

# Rapport

# Qualité programmation

---

**Écrit par :** CARRASCO Eliot, CHAABAN Bilal, GHERARDI Sylvain et HINDI Hassane

**Année du cursus universitaire :** 2017-2018



# Sommaire

Introduction.....	4
1 Organisation du travail .....	5
1.1 Outils utilisés .....	5
1.2 Répartition du travail.....	5
2 Implémentation.....	6
2.1 Architecture.....	6
2.2 Fonctionnalités .....	7
2.2.1 L'interface.....	7
2.2.2 Maquette.....	9
2.2.3 Formation .....	9
2.2.4 Unité d'enseignement .....	10
2.2.5 Unité d'enseignement à choix.....	10
3 Evolution du projet.....	11
3.1 Remaniements.....	11
3.2 Tests unitaires .....	12
3.2.1 Maquette.....	12
3.2.2 UEchoix.....	13
3.2.3 UEComposée .....	13
3.2.4 UEseule .....	14
3.2.5 Points à améliorer .....	14
4 Conclusion .....	15

## Introduction

Après avoir étudié le fonctionnement des tests unitaires, ainsi que les bonnes manières nécessaires à l'écriture d'un code propre et compréhensible, il nous a été demandé de créer un groupe de quatre personnes, notre groupe s'est donc constitué de CARRASCO Eliot, HINDI Hassane, GHERARDI Sylvain et de CHAABAN Bilal. Il nous a été demandé par la suite de réaliser un projet proposé par le professeur. Le but du projet durant ces deux mois était de développer grâce au langage C++ une solution permettant au personnel de la FST, de créer et de gérer plus facilement les maquettes universitaires, tout en respectant les contraintes imposées par les consignes.

Il s'agit par ailleurs du premier projet de qualité de programmation que nous réalisons, ce qui nous permet de prendre en main les outils de tests nécessaires pour réaliser un code propre et durable.

# 1 Organisation du travail

Après avoir découvert le sujet de notre projet et avant de commencer la programmation, nous avons commencé à mettre en place les différents outils informatiques permettant de faciliter et d'optimiser le travail en groupe.

## 1.1 Outils utilisés

Dans un premier temps nous avons décidé d'utiliser une plateforme permettant de faciliter la communication ainsi que le partage de documents entre membres du groupe, pour cela nous nous sommes orientés vers l'application professionnelle appelée « Discord » permettant de communiquer textuellement et de partager nos avancements sur le sujet du projet.

Nous avons ensuite décidé de créer un dépôt sur l'outil de *versionning* Git permettant à tous les membres du groupe de programmer en simultané sur le projet, optimisant ainsi notre travail.

En ce qui concerne notre environnement de développement, nous avons tous utilisé *CodeBlocks*.

Pour finir il était nécessaire de choisir un Framework nous permettant de réaliser les tests unitaires, notre choix s'est donc porté sur *Catch2*.

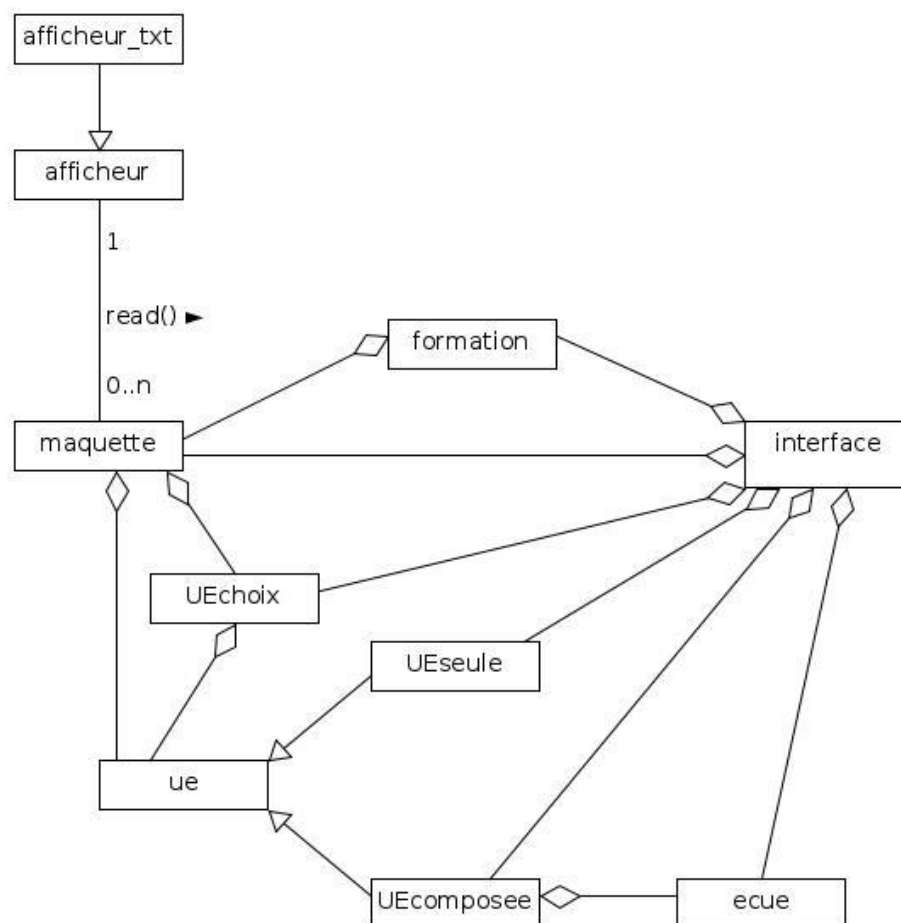
## 1.2 Répartition du travail

Notre groupe a eu l'idée de partager les tests unitaires par classe présente dans l'uml réalisé au préalable, CARRASCO Eliot s'est donc occupé de la classe « *UeComposée* », HINDI Hassane des classes « *ecue* » ainsi que « *UEchoix* », CHAABAN Bilal de « *UEseule* » et pour finir GHERARDI Sylvain s'est occupé des classes « *maquette* » et « *formation* ».

## 2 Implémentation

### 2.1 Architecture

Il était nécessaire, avant de débiter la programmation, d'organiser l'architecture de notre programme, en prévoyant les classes qui seront créées et les différents liens possibles entre elles.



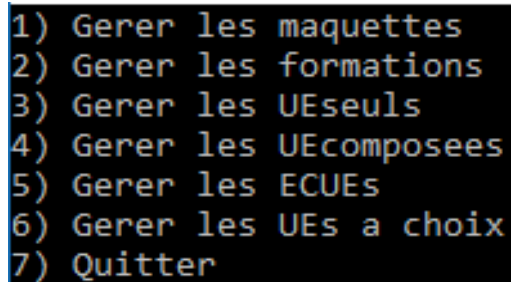
Dans cette architecture, la structure composée des classes formations, maquette, UEchoix, ue, UEseule, UEcomposee et ecue permet de stocker n'importe quel type de maquette et sont indépendantes du reste du programme.

Autour de cette structure nous avons ajouté les classes interface, afficheur et afficheur\_txt qui permettent à l'utilisateur d'interagir avec cette structure.

## 2.2 Fonctionnalités

### 2.2.1 L'interface

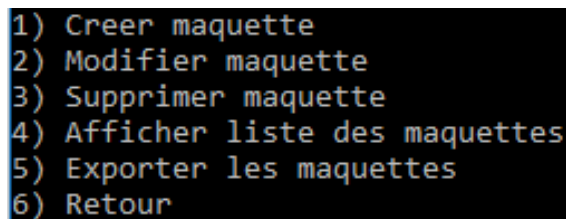
L'interface va permettre ici l'interaction directe entre le l'utilisateur et le programme, l'interface va être constitué de plusieurs menus, permettant la gestion des différentes unités d'enseignements. L'utilisateur pourra ainsi naviguer entre les différents menus présents, gérer entièrement les maquettes, les différents types d'unités d'enseignement ou encore les formations.

A screenshot of a terminal window showing a menu with seven options. The text is as follows:

```
1) Gerer les maquettes
2) Gerer les formations
3) Gerer les UEseuls
4) Gerer les UEcomposees
5) Gerer les ECUEs
6) Gerer les UEs a choix
7) Quitter
```

*Figure 1 : Premier menu proposé à l'utilisateur*

#### 2.2.1.1 Gestion des maquettes

A screenshot of a terminal window showing a menu with six options. The text is as follows:

```
1) Creer maquette
2) Modifier maquette
3) Supprimer maquette
4) Afficher liste des maquettes
5) Exporter les maquettes
6) Retour
```

*Figure 2 : Menu pour la gestion d'une maquette*

On retrouve les actions de base pour travailler sur une maquette :

- Le menu modifier permet de modifier l'entête, ajouter et supprimer des ues.
- Le menu exporter permet de créer des fichiers textes en fonctions des modèles préparés dans les classes dérivées de "afficheur".

Tous les menus suivants suivent la même logique que celui-ci.

#### 2.2.1.2 Gestion des formations

```
1) Creer formation
2) Modifier formation
3) Supprimer formation
4) Afficher toutes les formation
5) Retour
```

*Figure 3: Menu pour la gestion des formations*

#### 2.2.1.3 Gestion des unités d'enseignements seules

```
1) Creer UE seule
2) Modifier UE seule
3) Supprimer UE seule
4) Afficher la liste des UE seules
5) Retour
```

*Figure 4 : Menu pour la gestion des UE seules*

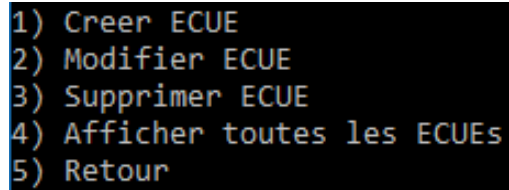
#### 2.2.1.4 Gestion des unités d'enseignements composées

```
1) Creer UE composee
2) Modifier UE composee
3) Supprimer UE composee
4) Afficher la liste des UEs composees
5) Retour
```

*Figure 5: Menu pour la gestion des UE composées*



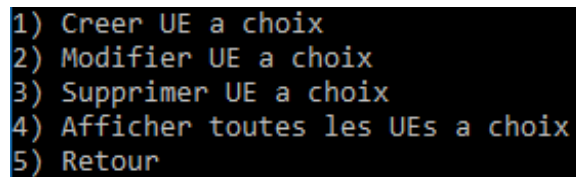
### 2.2.1.5 Gestion des éléments constitutifs d'une unité d'enseignement



```
1) Creer ECUE
2) Modifier ECUE
3) Supprimer ECUE
4) Afficher toutes les ECUES
5) Retour
```

Figure 6 : Menu pour la gestion des ECUES

### 2.2.1.6 Gestion des unités d'enseignement



```
1) Creer UE a choix
2) Modifier UE a choix
3) Supprimer UE a choix
4) Afficher toutes les UEs a choix
5) Retour
```

Figure 7: menu pour la gestion d'UE a choix

## 2.2.2 Maquette

La maquette va organiser les unités d'enseignements, quel qu'en soit le type. La classe maquette va donc contenir un domaine, le parcours ainsi que l'année et le semestre en cours. De plus, il contiendra des vecteurs de pointeurs d'unité d'enseignement et d'unités d'enseignement à choix. Pour finir, un attribut « position\_ue\_choix\_dans\_ue » permettra d'organiser correctement les unités d'enseignements ainsi que les unités d'enseignements à choix.

## 2.2.3 Formation

L'utilisateur aura la possibilité de créer une formation, contenant des maquettes, détaillées ci-dessus. Elle contiendra aussi une mention.

## 2.2.4 Unité d'enseignement

La classe Unité d'enseignement « ue » correspond à une classe abstraite, les classes « UEseule » et « UEcomposee » vont alors hériter d' « ue ».

### 2.2.4.1 Unité d'enseignement seule

L'unité d'enseignement seule va être représentée par les heures de cours magistraux, de travaux dirigés et de travaux pratiques, d'un code, d'un intitulé, des crédits qu'elle procure ainsi que du coefficient.

### 2.2.4.2 Unité d'enseignement composée

L'unité d'enseignement composée va être représentée d'un code, d'un intitulé, des crédits qu'elle procure ainsi que du coefficient. Elle aura aussi la particularité de contenir un vecteur de pointeurs d'ecue.

## 2.2.5 Unité d'enseignement à choix

L'unité d'enseignement à choix va tout simplement correspondre à un vecteur de pointeurs d'unités d'enseignements. L'utilisateur aura donc le choix entre différentes unités d'enseignement, ce qui définira l'ue à choix.

## 3 Evolution du projet

### 3.1 Remaniements

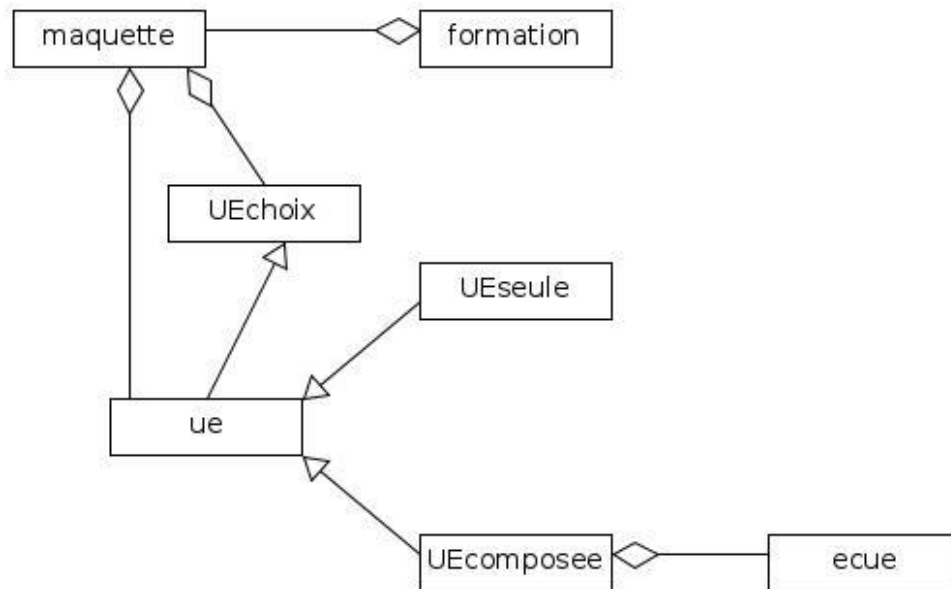


Figure 8 : UML de départ du projet

Initialement, ue était une fille de UEchoix, mais ue étant une classe abstraite, il était impossible de sauvegarder des ue et UEchoix et de les différencier dans maquette, il y avait un problème d'object slicing. C'est pour cela que nous avons décidé que l'UEchoix comporte des ue.

## 3.2 Tests unitaires

### 3.2.1 Maquette

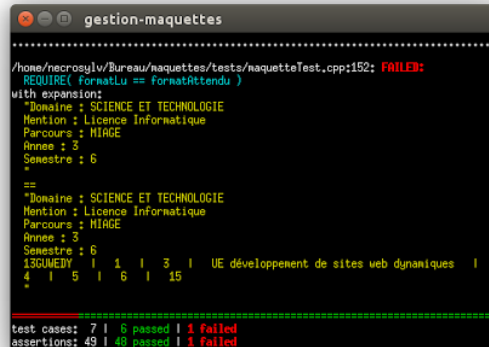
```
SECTION("Une maquette avec UEchoix s'affiche correctement")
{
    formatAttendu += code + " | " + std::to_string(coefficient) + " | " + std::to_string(credits) + " | " + libele + " | " +
        + std::to_string(heures_cm) + " | " + std::to_string(heures_td) + " | " + std::to_string(heures_tp) + " | " + std::to_string(heures_totales) + "\n";

    liste_ue_choix.push_back(new UEchoix());
    liste_ue_choix[0]->ajouter_ue(new UEseule{heures_cm, heures_td, heures_tp, code, libele, credits, coefficient});
    maquette maquette_avec_ue_choix{liste_ue, liste_ue_choix, mention, parcours, annee, semestre};

    std::ostringstream ost;
    ost << maquette_avec_ue_choix;
    formatLu = ost.str();

    REQUIRE(formatLu == formatAttendu);

    for(unsigned int i=0; i<liste_ue_choix.size(); i++)
        delete liste_ue_choix[i];
}
```



```
gestion-maquettes
/home/necrosylv/Bureau/maquettes/tests/maquetteTest.cpp:152: FAILED:
  REQUIRE( formatLu == formatAttendu )
with expansion:
  "Domaine : SCIENCE ET TECHNOLOGIE
  Mention : Licence Informatique
  Parcours : MIRAGE
  Année : 3
  Semestre : 6
  "
  ==
  "Domaine : SCIENCE ET TECHNOLOGIE
  Mention : Licence Informatique
  Parcours : MIRAGE
  Année : 3
  Semestre : 6
  13GUNEEDY | 1 | 3 | UE développement de sites web dynamiques |
  4 | 5 | 6 | 15
  "
test cases: 7 | 6 passed | 1 failed
assertions: 49 | 48 passed | 1 failed
```

Figure 9 : Test de l'affichage d'une maquette avec UEchoix

On peut voir qu'une maquette qui contient uniquement une UEchoix ne s'affiche pas correctement.

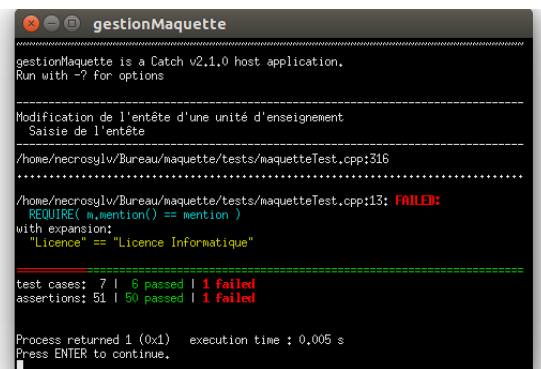
```
TEST_CASE("Modification de l'entête d'une unité d'enseignement", "[maquette]")
{
    std::string domaine = "SCIENCE ET TECHNOLOGIE";
    std::string mention = "Licence Informatique";
    std::string parcours = "MIRAGE";
    unsigned int annee = 3;
    unsigned int semestre = 6;

    SECTION("Saisie de l'entête")
    {
        maquette maquette_default{};

        std::string formatLu = mention + "\n" + parcours + "\n" + std::to_string(annee)
            + "\n" + std::to_string(semestre) + "\n";
        std::istringstream ist{formatLu};
        std::ostringstream ost{};

        maquette_default.saisir_maquette(ost, ist);

        valeursEntete(maquette_default, domaine, mention, parcours, annee, semestre);
    }
}
```



```
gestionMaquette
gestionMaquette is a Catch v2.1.0 host application.
Run with -? for options

Modification de l'entête d'une unité d'enseignement
Saisie de l'entête

/home/necrosylv/Bureau/maquette/tests/maquetteTest.cpp:316
/home/necrosylv/Bureau/maquette/tests/maquetteTest.cpp:13: FAILED:
  REQUIRE( m.mention() == mention )
with expansion:
  "Licence" == "Licence Informatique"
test cases: 7 | 6 passed | 1 failed
assertions: 51 | 50 passed | 1 failed

Process returned 1 (0x1)   execution time : 0.005 s
Press ENTER to continue.
```

Figure 10 : Test modification de l'entête d'UE

Grâce à ce test on remarque que la modification de l'entête d'une maquette pose problème lorsque le libellé de la mention ou du parcours s'écrit en plusieurs mots. En

effet, l'espace est considéré comme un séparateur et "Licence Informatique" remplit donc deux paramètres à la fois.

Outre ces deux principaux problèmes les tests sur cette classe ont permis de corriger une erreur lors de l'initialisation du paramètre `d_position_ue_choix_dans_ue` et de modifier des problèmes de style dans la fonction d'affichage.

### 3.2.2 UEchoix

Sur cette image, on peut voir que les tests de la classe UEchoix ont permis de mettre en évidence une petite erreur de redimensionnement du vecteur d'UEs. Lorsque l'utilisateur cherche à supprimer une UE de l'UE à choix, la fonction ne redimensionnait pas correctement le vecteur. En effet, ce dernier était redimensionné à la même taille qu'avant la suppression.

```

void UEchoix::supprimer_ue(ue *u)
{
    unsigned int i = 0;
    while(u != d_liste_ue[i])
        i++;
    delete d_liste_ue[i];
    for(; i < d_liste_ue.size()-1; i++)
        std::swap(d_liste_ue[i], d_liste_ue[i+1]);
    d_liste_ue.resize(d_liste_ue.size());
}

std::vector<ue*> UEchoix::liste_ue() const
{
    return d_liste_ue;
}

```

```

La suppression d'une UE est correct
F:\Hassane\Cours\Fac\L3\Qualite de programmation\Gestion-Maquettes\tests\UEchoix
Test.cpp:103
.....
F:\Hassane\Cours\Fac\L3\Qualite de programmation\Gestion-Maquettes\tests\UEchoix
Test.cpp:120: FAILED:
  REQUIRE( uec.nombre_ue_choix() == 0 )
with expansion:
  1 == 0
-----
test cases: 20 | 19 passed | 1 failed
assertions: 118 | 117 passed | 1 failed
Process returned 1 (0x1)   execution time : 0.047 s
Press any key to continue.

```

Figure 11: Suppression d'une UE choix

### 3.2.3 UEComposée

```

SECTION("La lecture est correcte")
{
    std::ostringstream ost{};
    std::string formatLu = nouveau_code + " "
        + nouvel_intitule + " "
        + nouveaux_credits + " "
        + nouveau_coefficient;

    std::istringstream ist{formatLu};
    uec.modifier_entete(ost, ist);

    REQUIRE(uec.code() == nouveau_code);
    REQUIRE(uec.intitule() == nouvel_intitule);
    REQUIRE(std::to_string(uec.credits()) == nouveaux_credits);
    REQUIRE(std::to_string(uec.coefficient()) == nouveau_coefficient);
}

```

```

Entrées sorties
La lecture est correcte
/home/necrosylv/Bureau/maquette/tests/UEComposéeTest.cpp:139
.....
/home/necrosylv/Bureau/maquette/tests/UEComposéeTest.cpp:151: FAILED:
  REQUIRE( uec.intitule() == nouvel_intitule )
with expansion:
  "UE" == "UE Intelligence Artificielle"
-----
test cases: 24 | 23 passed | 1 failed
assertions: 150 | 149 passed | 1 failed
Process returned 1 (0x1)   execution time : 0.006 s
Press ENTER to continue.

```

Figure 12 : Lecture d'une UE composée



## 4 Conclusion

Tout au long de ce projet nous avons été surpris par la capacité des tests unitaires à permettre de repérer des erreurs dans notre code qui auraient été particulièrement compliquées à détecter sans cela. Nous avons pu en corriger une majorité, mais malheureusement d'autres modifications qui auraient dû être faites sont restées en suspens.

Concernant les nouvelles notions de cours qui nous ont été dispensées, ce sont vraiment les tests unitaires qui ont été d'une plus grande aide dans ce projet. Quelques conseils pour coder en groupe auraient cependant été les bienvenus, en effet nous avons déjà eu des cours de gestion de projet pour travailler à plusieurs, mais jamais sur un projet de programmation qui peut être sauvegardé sur un git. Tous les membres d'un groupe n'ayant pas les mêmes idées pour organiser un git il a parfois été difficile de s'y retrouver.

