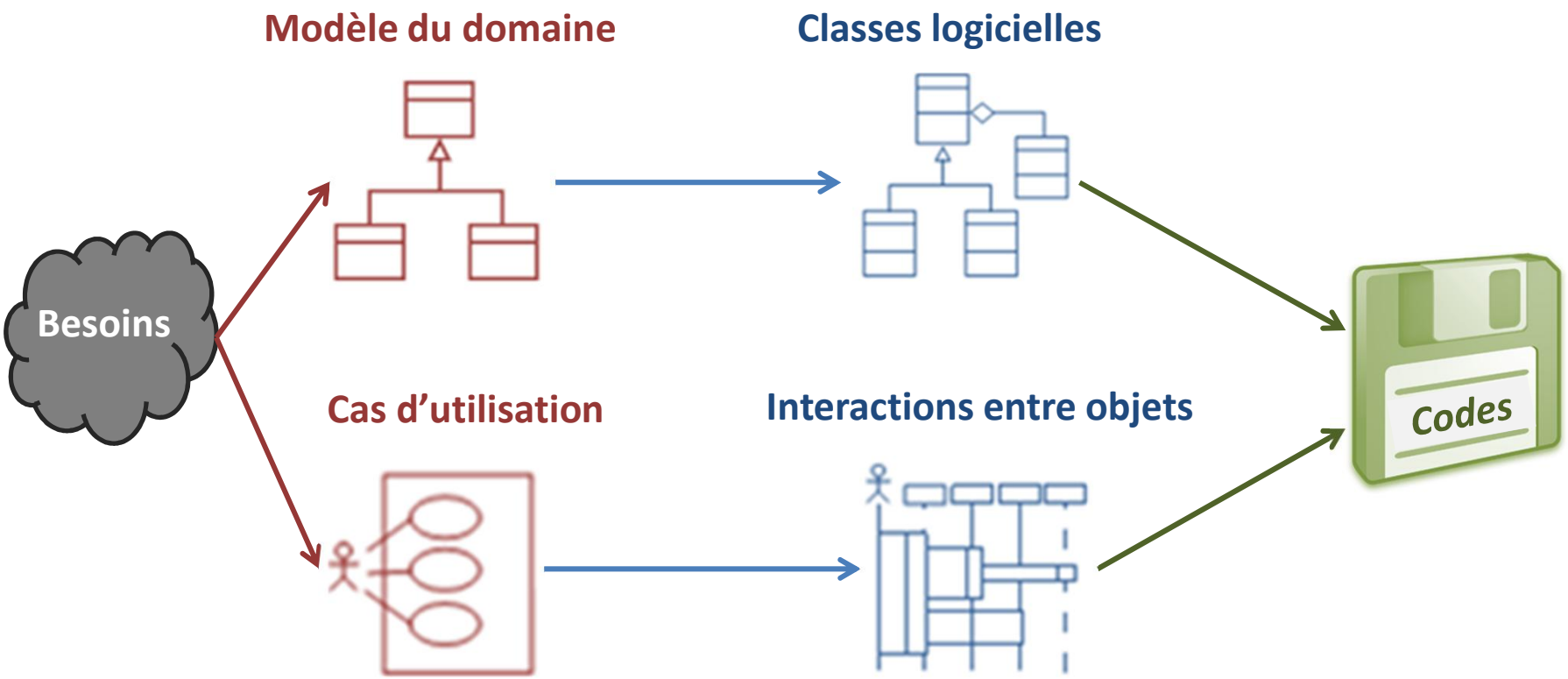
Génie logiciel 1

Développement piloté par les tests

Vue globale du développement

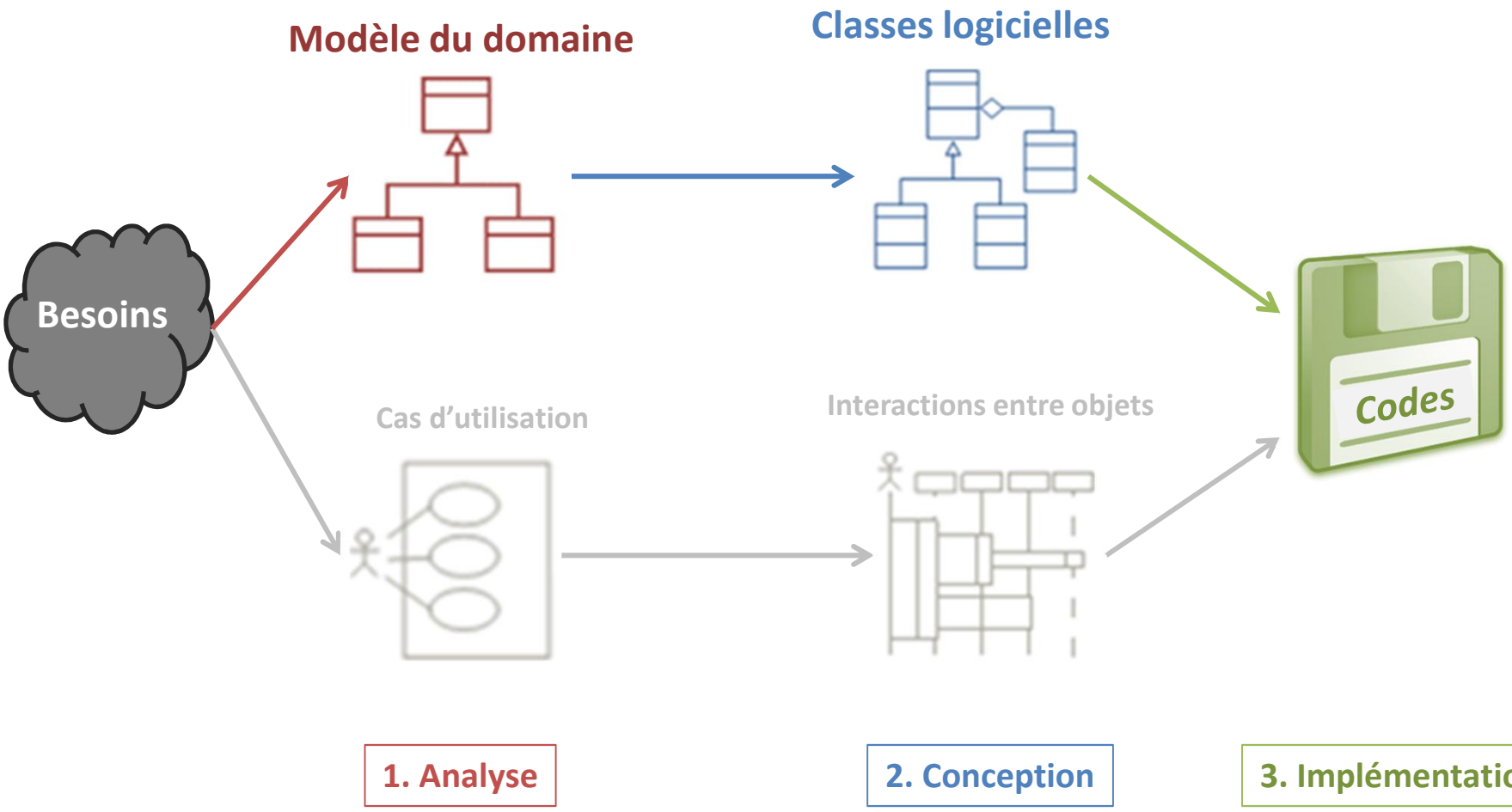


1. Analyse
Découverte des concepts fondamentaux
Identification des fonctionnalités

2. Conception
Spécification de classes et objets
Affectation de comportements aux objets

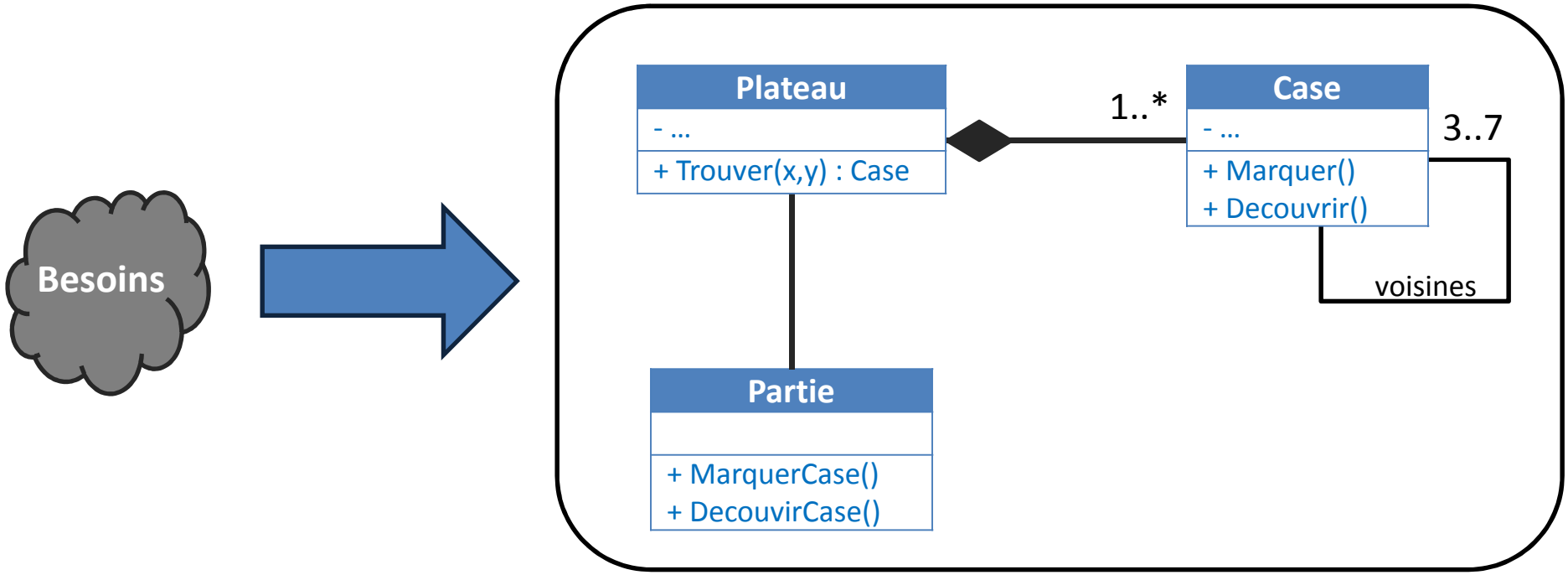
3. Implémentation
Codage + tests unitaires
Tests d'intégration

Modélisation statique

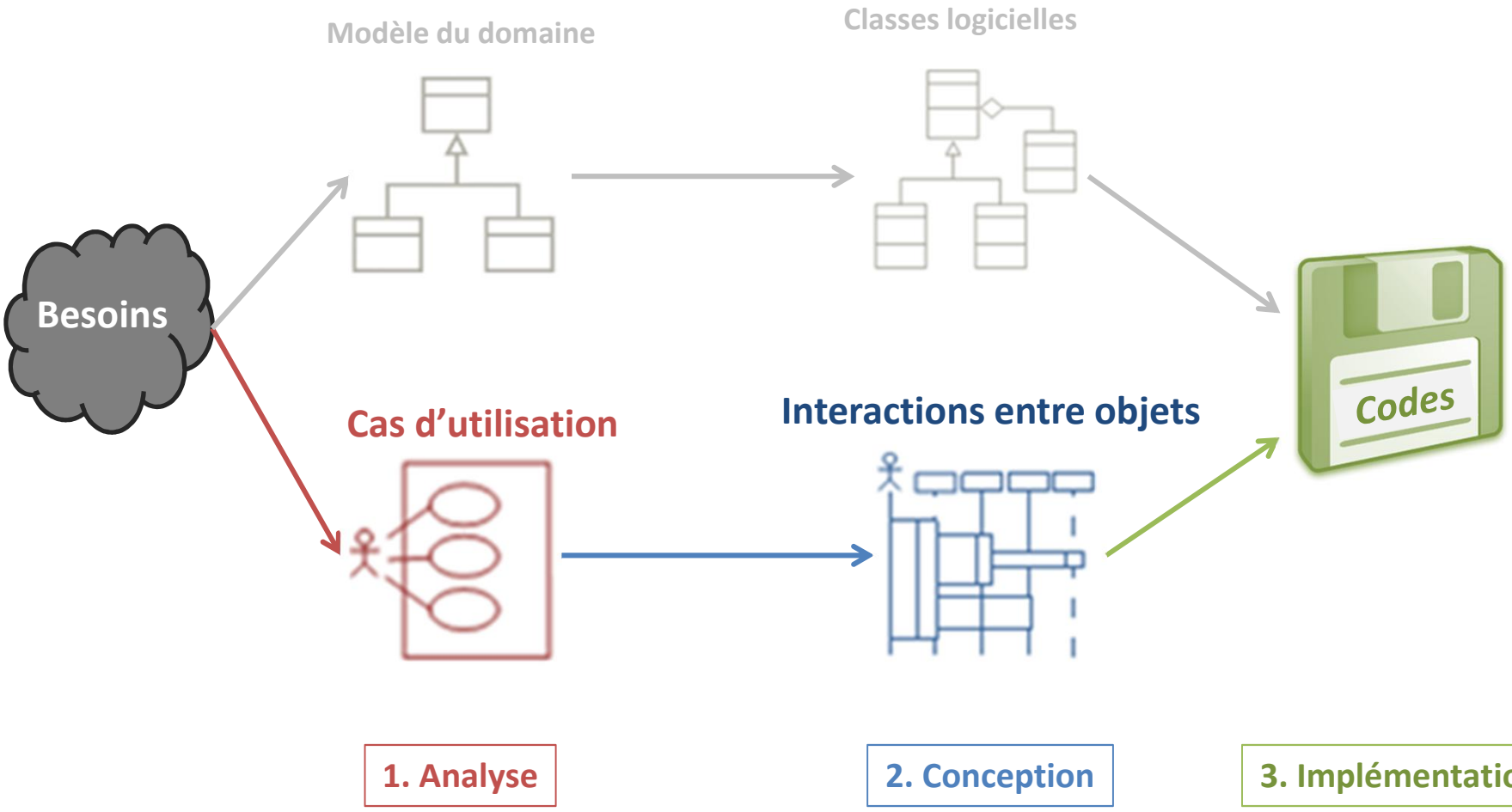


Modélisation statique

- **Objectif** → définir les classes, les attributs et les méthodes
  - *Quels sont les briques fondamentales ?* → ***Classes logicielles***
  - *Quelles sont les propriétés significatives ?* → ***Attributs et Associations***
  - *Quelles sont les comportements notables ?* → ***Méthodes***

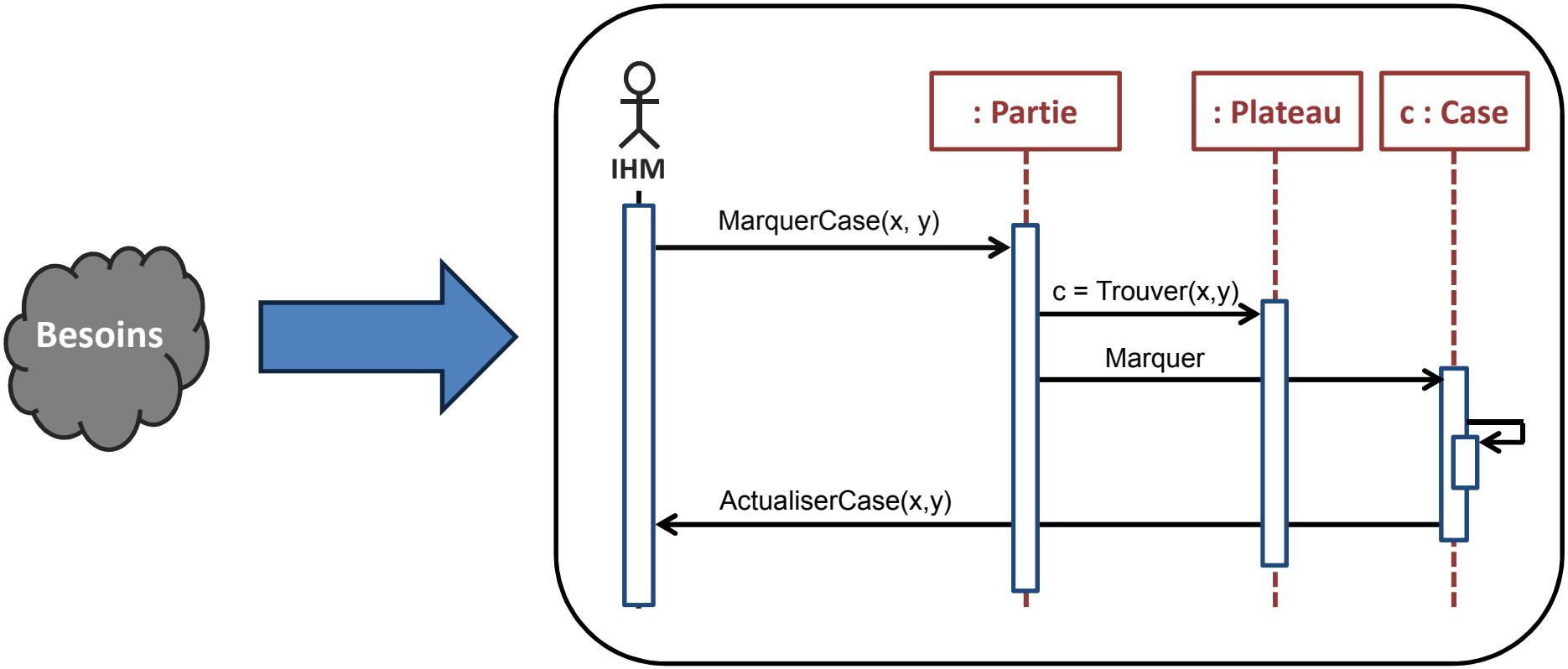


Modélisation dynamique



Modélisation dynamique

- **Objectif** → définir la manière dont les objets réalisent une fonctionnalité
  - *Comment collaborent les objets ?* → ***Echange de messages***
  - *Quels sont les messages échangés ?* → ***Méthodes***



Comment créer un modèle dynamique ?

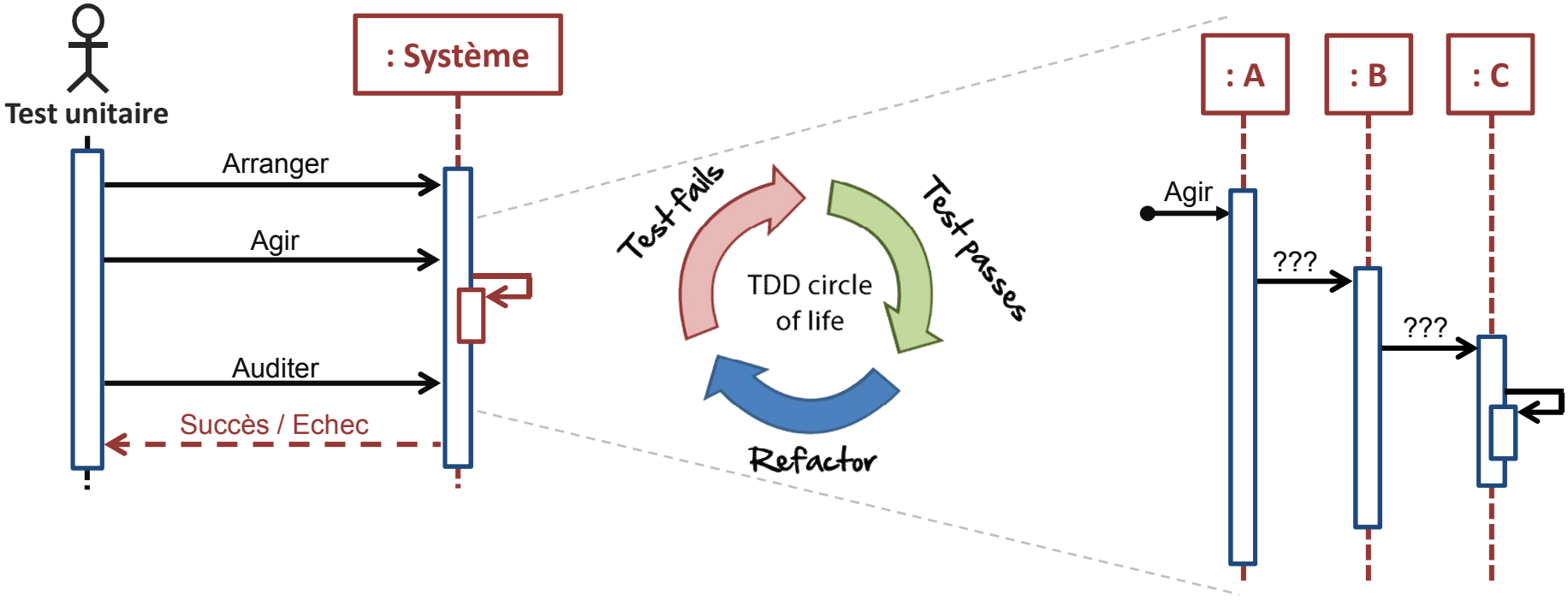
- Le TDD est une technique agile de modélisation dynamique

1. REDIGER UN TEST UNITAIRE

(Spécification << en boite noire >> d'une fonctionnalité)

2. IMPLEMENTER LE CODE POUR REUSSIR LE TEST

(Conception et mise en ouvre d'un modèle dynamique)



Génie Logiciel 1

Exercices

Ex. 2.7 – Tests unitaires

- Implémentez une fonction qui compte le nombre de chiffres utiles dans un entier non négatif (7 a une chiffre, 14 en a deux, etc).

```
void TestValeur(long n, int vrai)
{
  Assert.AreEqual(CompterChiffres(n), vrai);
}

public void Test0() {
  TestValeur(0, 1);
}

public void Test9() {
  TestValeur(9, 1);
}
```

```
public void Test10() {
  TestValeur(10, 2);
}

public void Test1456() {
  TestValeur(1456, 4);
}


public void TestMax() {
  TestValeur(Int64.MaxValue, 19);
}
```

Ex. 2.8 – Micro-transformations

- Implémentez une fonction qui calcule la représentation romaine d'un entier entre 1 et 3999.

```
void TestValue(int n, string r) {
  Assert.AreEqual(ToRoman(n),r);
}

void Test1() {
  TestValue (1, "I");
}
void Test3() {
  TestValue(3, "III");
}

void Test5() {
  TestValue(5, "V");
}
void Test8() {
  TestValue(8, "VIII");
}
```

```
public void Test27() {
  TestValeur(27, "XXVII");
}

public void Test51() {
  TestValeur(51, "LI");
}

void Test4() {
  TestValue(4, "IV");
}
void Test9() {
  TestValue(9, "IX");
}
void Test40() {
  TestValue(8, "XL");
}
```

Ex. 2.9 – Arranger-Agir-Auditer

- Implémentez le jeu << chasse au Wumpus >>.
  - C'est une partie de cache-cache avec un monstre, le Wumpus.
  - Le terrain de jeu est un réseau de cavernes, chacune étant reliée à quatre autres.
  - Les cavernes présentent des pièges : les puits sans fond. Le joueur peut sentir la présence d'une piège si celle-ci se trouve dans l'une des salles adjacentes.
  - Le joueur peut sentir la présence du Wumpus s'il se trouve dans l'une des salles adjacentes à sa position.
  - Le joueur peut tirer une flèche dans l'une des salles adjacentes. Si le Wumpus n'est pas dans la salle cible, la flèche continue et traverse 4 salles.
  - La partie est gagnée si la flèche passe par la caverne du Wumpus.

Ex. 2.10 – Doublures de test

- Continuez l'Implémentation du jeu << chasse au Wumpus >>.
  - Les cavernes présentent un deuxième type de pièges, les chauves-souris, qui transportent le joueur dans une autre caverne sélectionnée aléatoirement.
  - Le joueur peut tirer une flèche dans l'une des cavernes adjacentes. Si le Wumpus n'est pas dans la caverne cible, la flèche continue jusqu'à traverser 4 cavernes **sélectionnées aléatoirement**.
  - Si la flèche ne passe pas par la caverne du Wumpus, ceci se déplace dans une caverne choisie au hasard parmi les quatre adjacentes à sa position.

Ex. 2.11 – Tests de bout en bout

- Terminez l'Implémentation du jeu << chasse au Wumpus >> en ajoutant les tests de bout en bouts suivants.
  - **Partie gagnée**. Le joueur explore quelques cavernes jusqu'à sentir la présence du Wumpus. Il tire une flèche vers la bonne direction et remporte la partie.
  - **Partie perdue 1**. Le joueur explore des cavernes jusqu'à tomber dans un puits.
  - **Partie perdue 2**. Le joueur explore quelques cavernes jusqu'à trouver les chauves-souris, qui transportent le jouer dans la caverne du Wumpus.
  - **Partie perdue 3**. Le joueur explore quelques cavernes jusqu'à sentir la présence du Wumpus. Il tir une flèche vers la mauvaise direction et le Wumpus se déplace dans la caverse où le jouer se trouve.
  - **Partie perdue 4**. Comme avant, mais c'est la flèche qui toue le joueur.