R504-Cycle de Vie d’un Projet Informatique

Enseignant : Morad M’LIK

Contexte Professionnel

Le professionnel RT peut être amené à gérer toutes les étapes d'un projet informatique depuis le cahier des charges jusqu'à la mise en production et la fourniture de documentation tant utilisateurs que technique.
Cette ressource a donc pour objectif de faire appréhender toutes les étapes d'un projet informatique depuis l'expression d'un besoin jusqu'au produit fini.

PARTIE 1 : Généralités

1.La Gestion « Old School »

Analyse du besoin aboutissant au lancement d'un projet de développement :

-- Définition des objectifs,

-- Rédaction du cahier des charges en spécifiant les attentes/contraintes fonctionnelles et techniques,

-- Choix de la stratégie de mise en production,

-- Prise en compte des problèmes de sécurité.

Cycle de gestion:

- La Cascade :
  - Point de départ, point d’arrivée
  - Diagramme de GANTT

- Cycle en V :
  - Cascade où chaque phase est accompagnée de tests

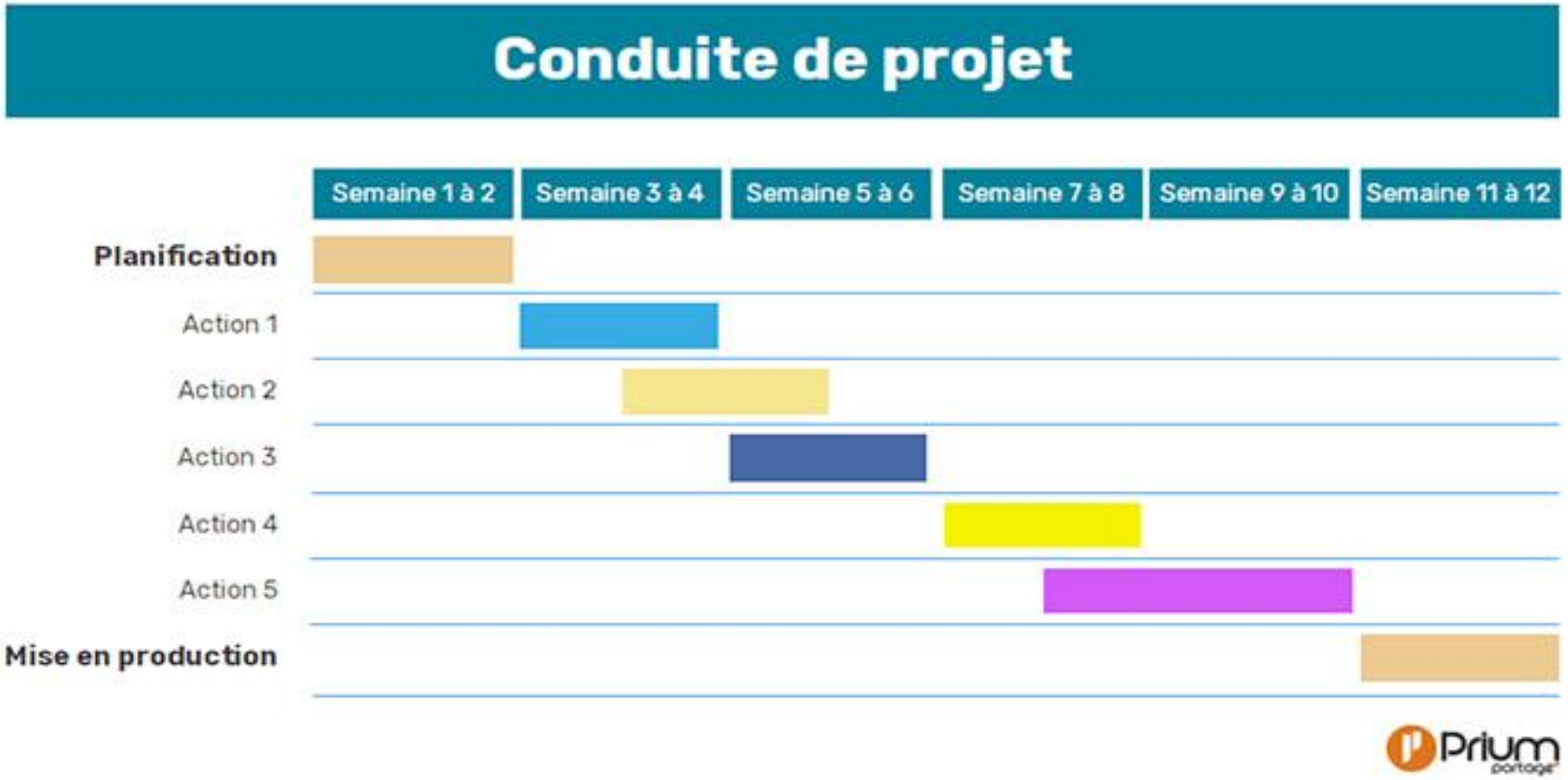
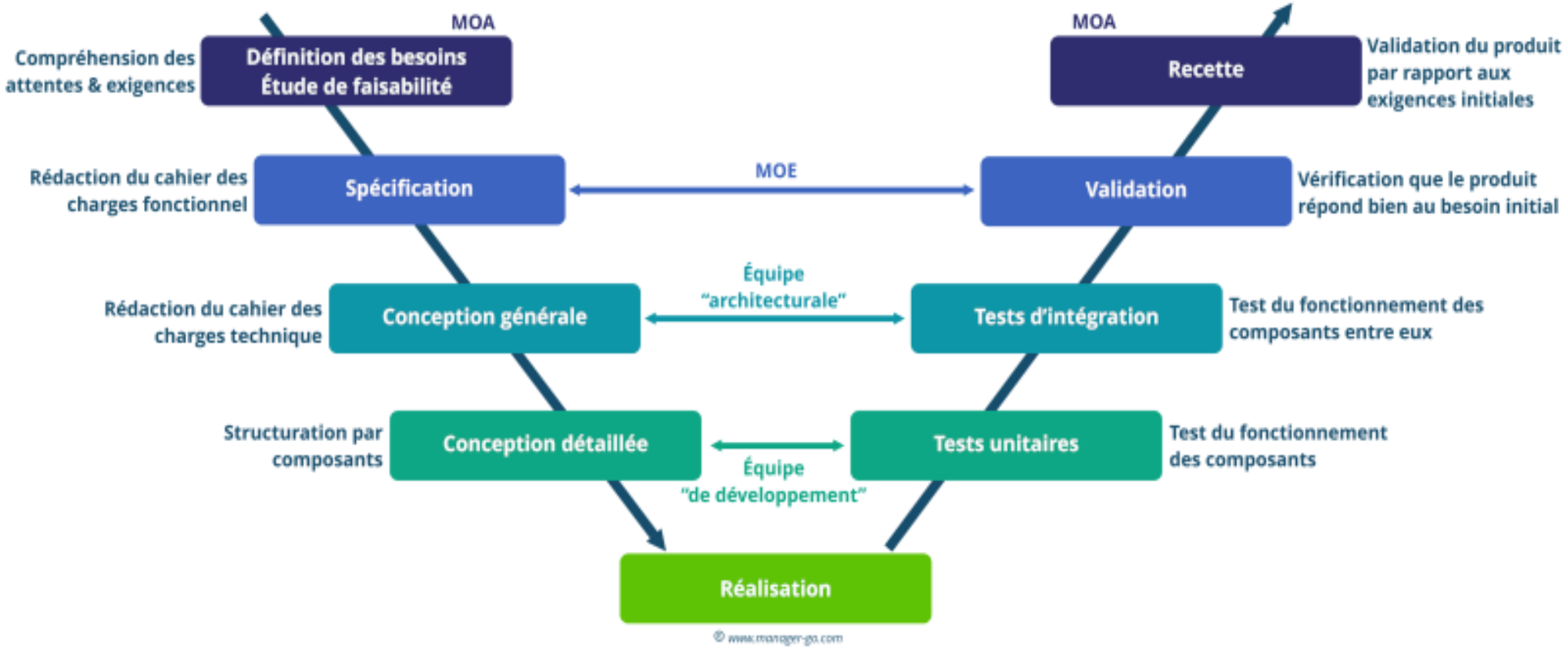


Diagramme de GANTT

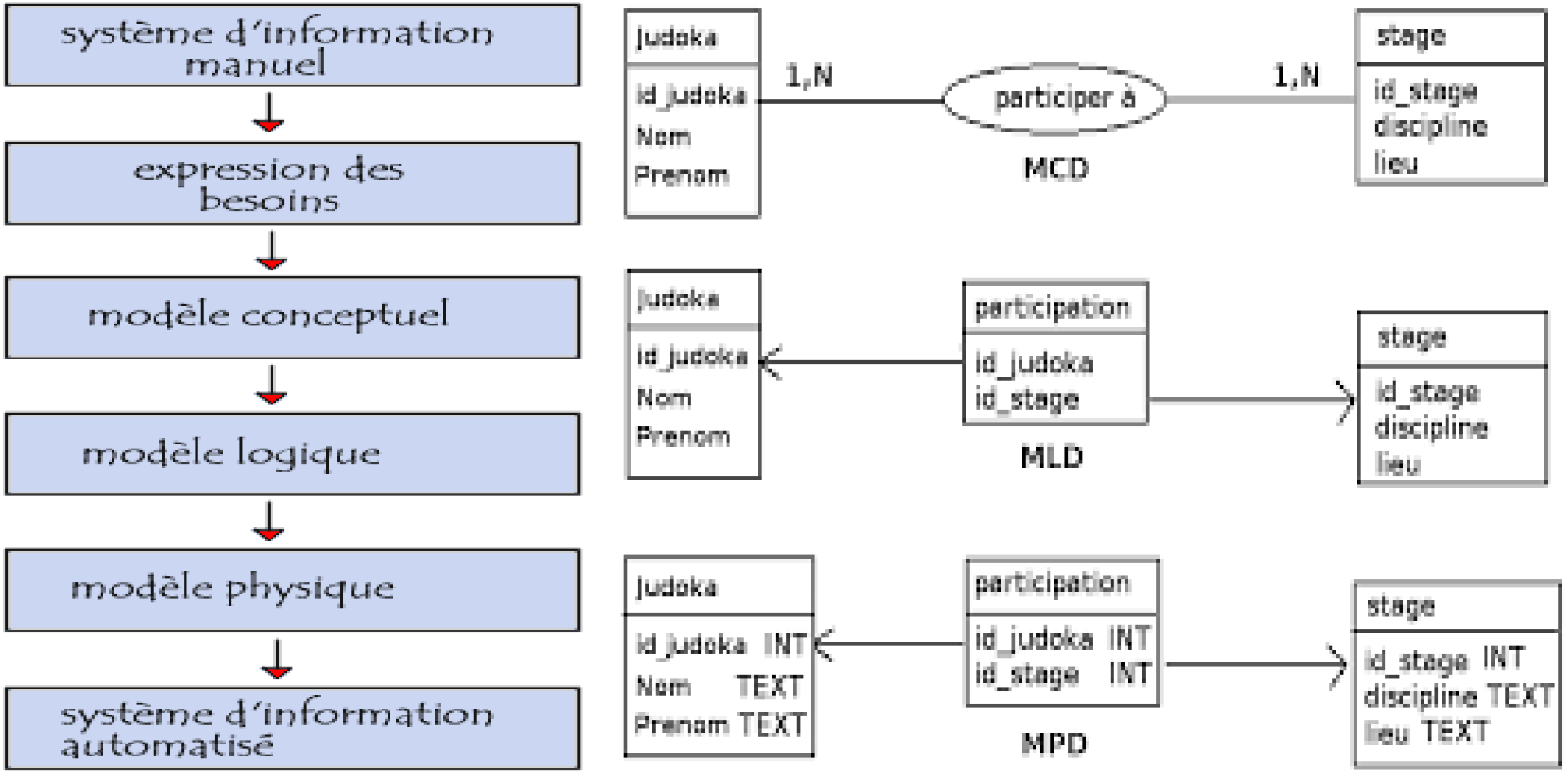

MOA
Compréhension des attentes & exigences
Définition des besoins
Étude de faisabilité
MOA
Recette
Validation du produit par rapport aux exigences initiales
Rédaction du cahier des charges fonctionnel
Spécification
MOE
Validation
Vérification que le produit répond bien au besoin initial
Équipe "architecturale"
Rédaction du cahier des charges technique
Conception générale
Tests d'intégration
Test du fonctionnement des composants entre eux
Structuration par composants
Conception détaillée
Équipe "de développement"
Tests unitaires
Test du fonctionnement des composants
Réalisation
© www.manager-go.com

Cycle en V

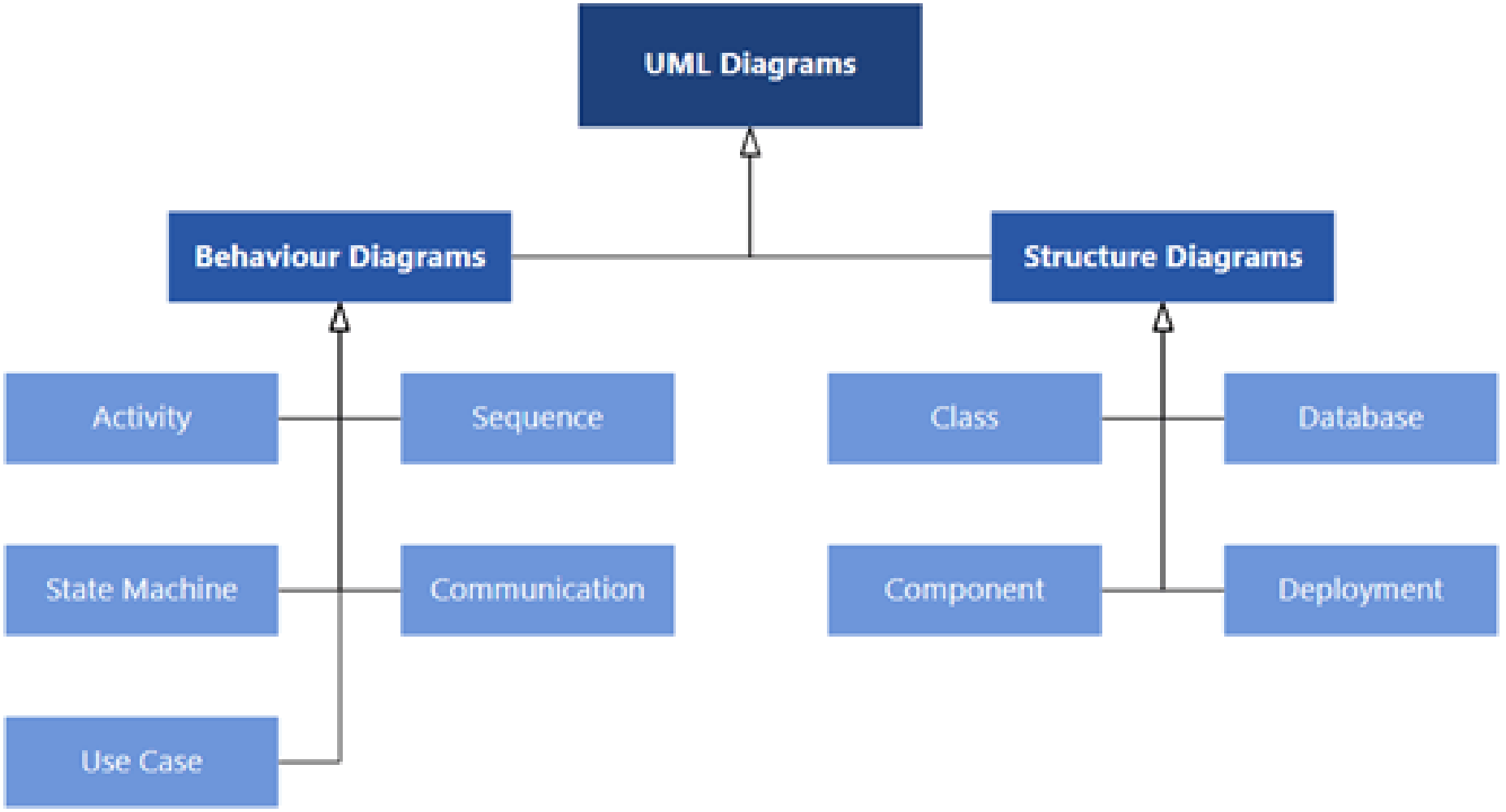
Outils de Conception:

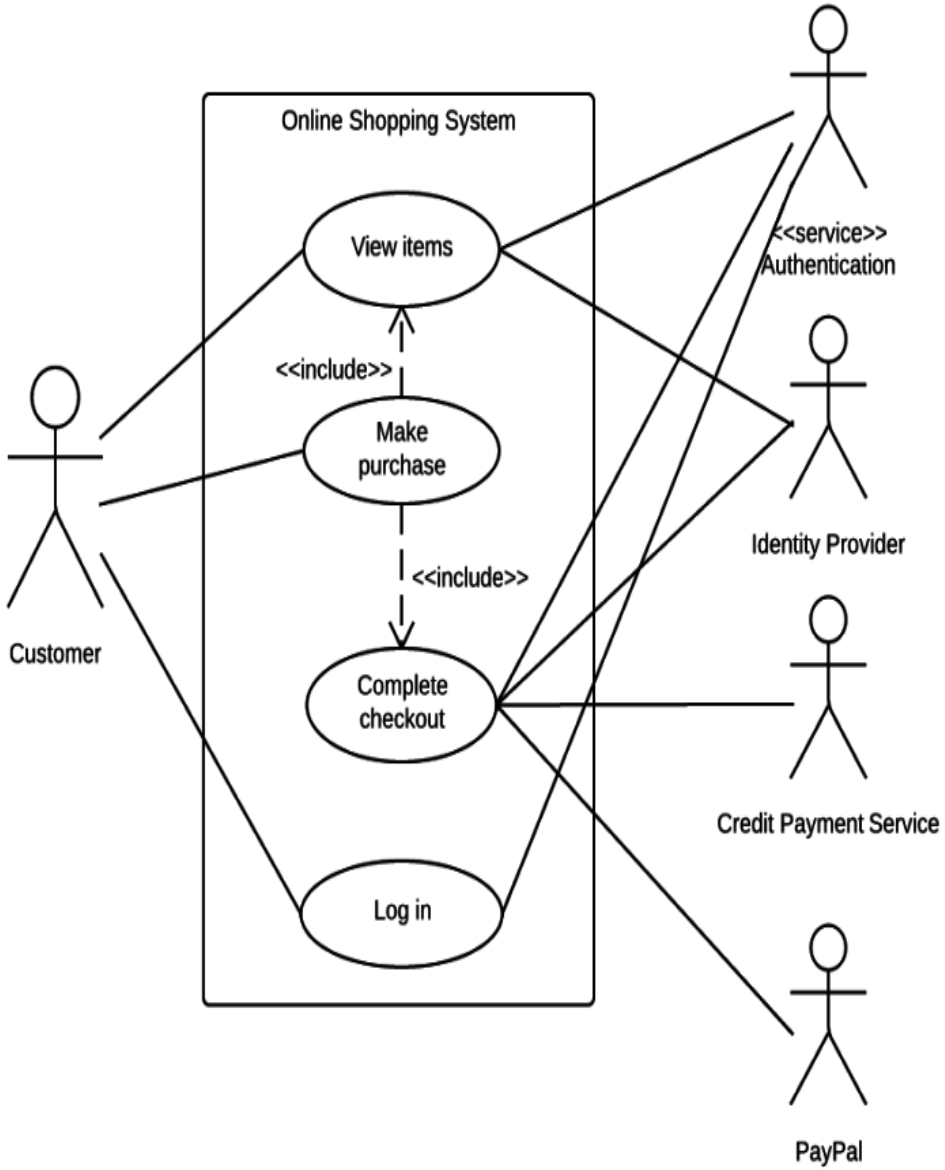
- MERISE :
  - Dédier à la conception de Bases de Données
  - Modèle Conceptuel, Logique et Physique de Données (MCD, MLD, MPD)

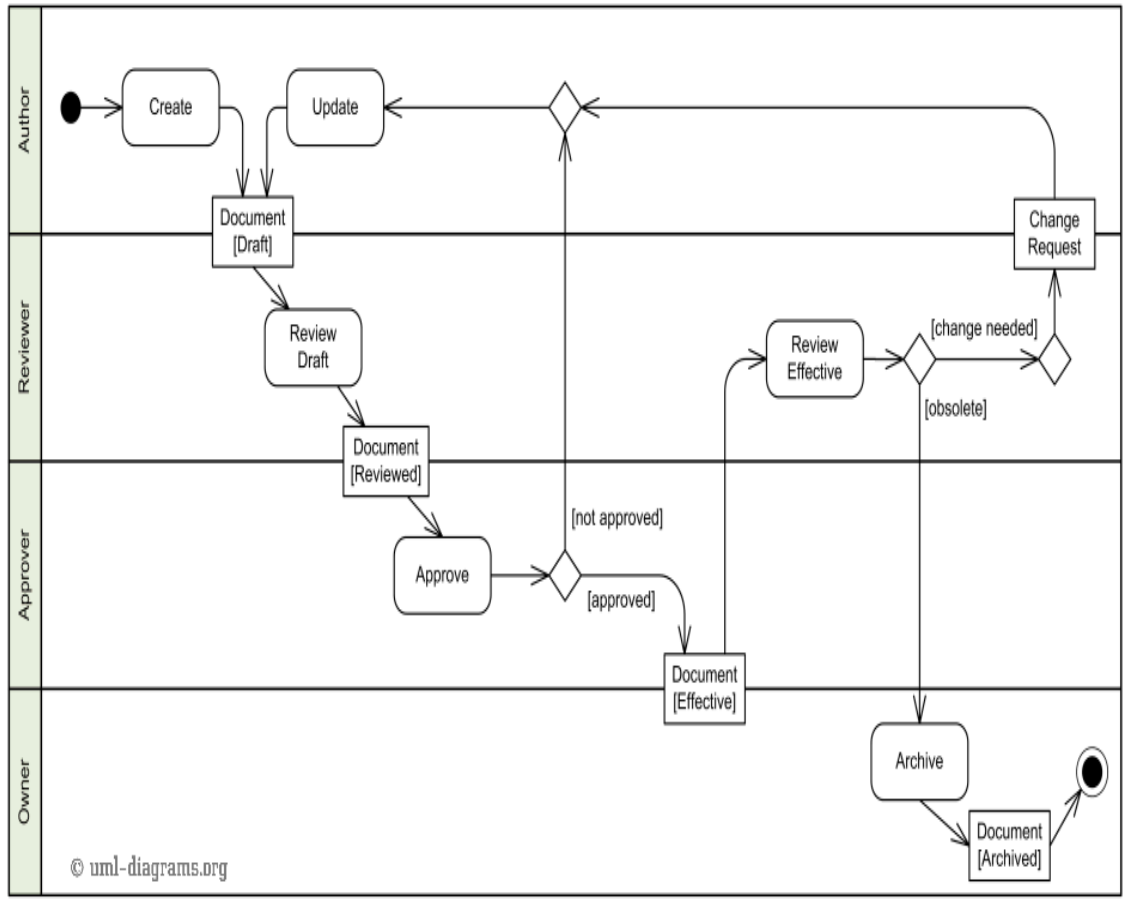
- UML :
  - Ensemble de 13 Diagrammes qui permettent de Conceptualiser tout projet
  - Diagram of Use Cases (DUC), Diagramme de Séquence, Diagramme de Classes, Diagramme d'activité et Diagramme de Composants y sont dédiés à l'informatique

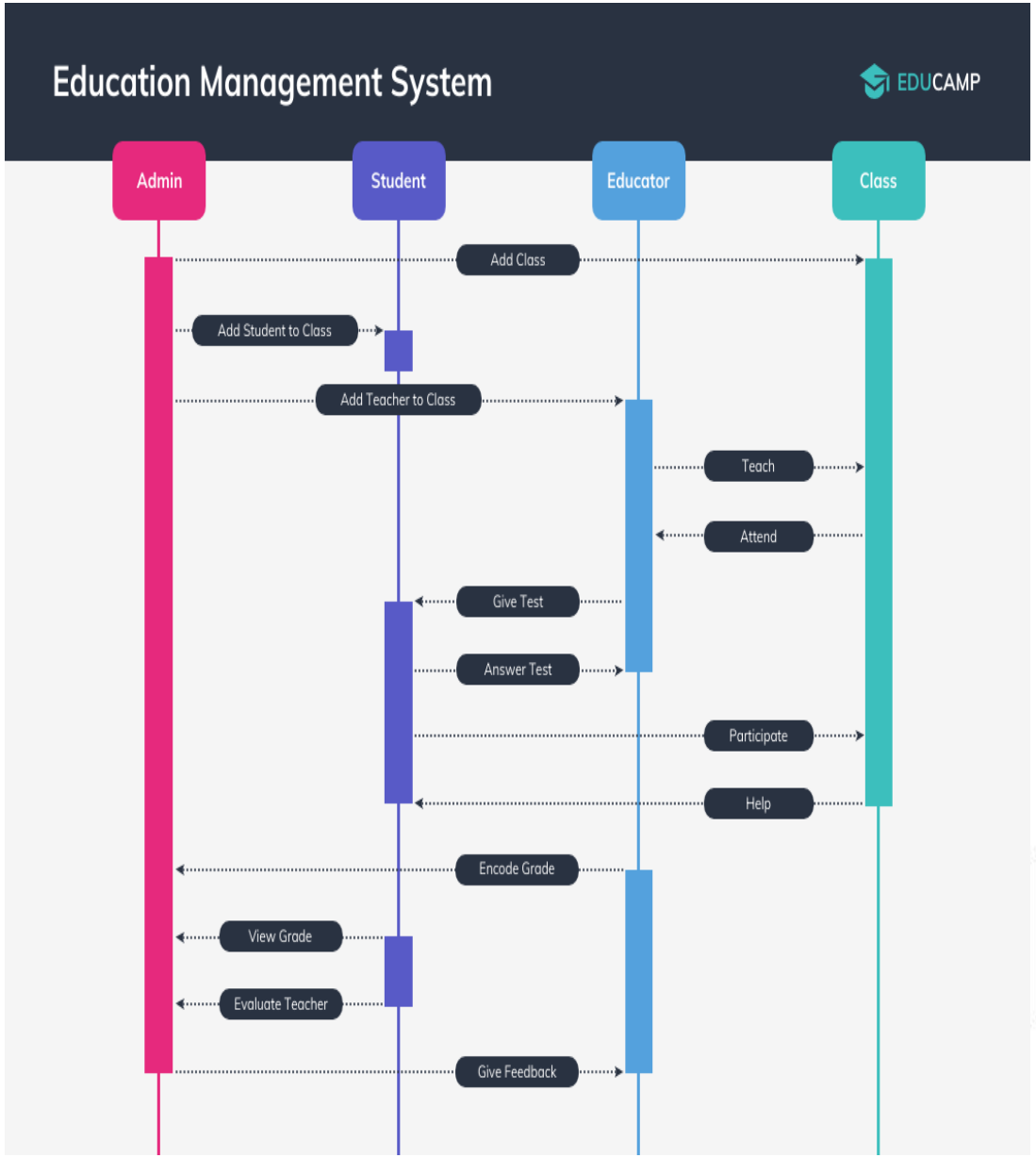

système d'information manuel
expression des besoins
modèle conceptuel
modèle logique
modèle physique
système d'information automatisé
Judoka
id_judoka
Nom
Prenom
1,N
participer à
1,N
stage
id_stage
discipline
lieu
MCD
Judoka
id_judoka
Nom
Prenom
participation
id_judoka
id_stage
stage
id_stage
discipline
lieu
MLD
Judoka
id_judoka INT
Nom TEXT
Prenom TEXT
participation
id_judoka INT
id_stage INT
stage
id_stage INT
discipline TEXT
lieu TEXT
MPD

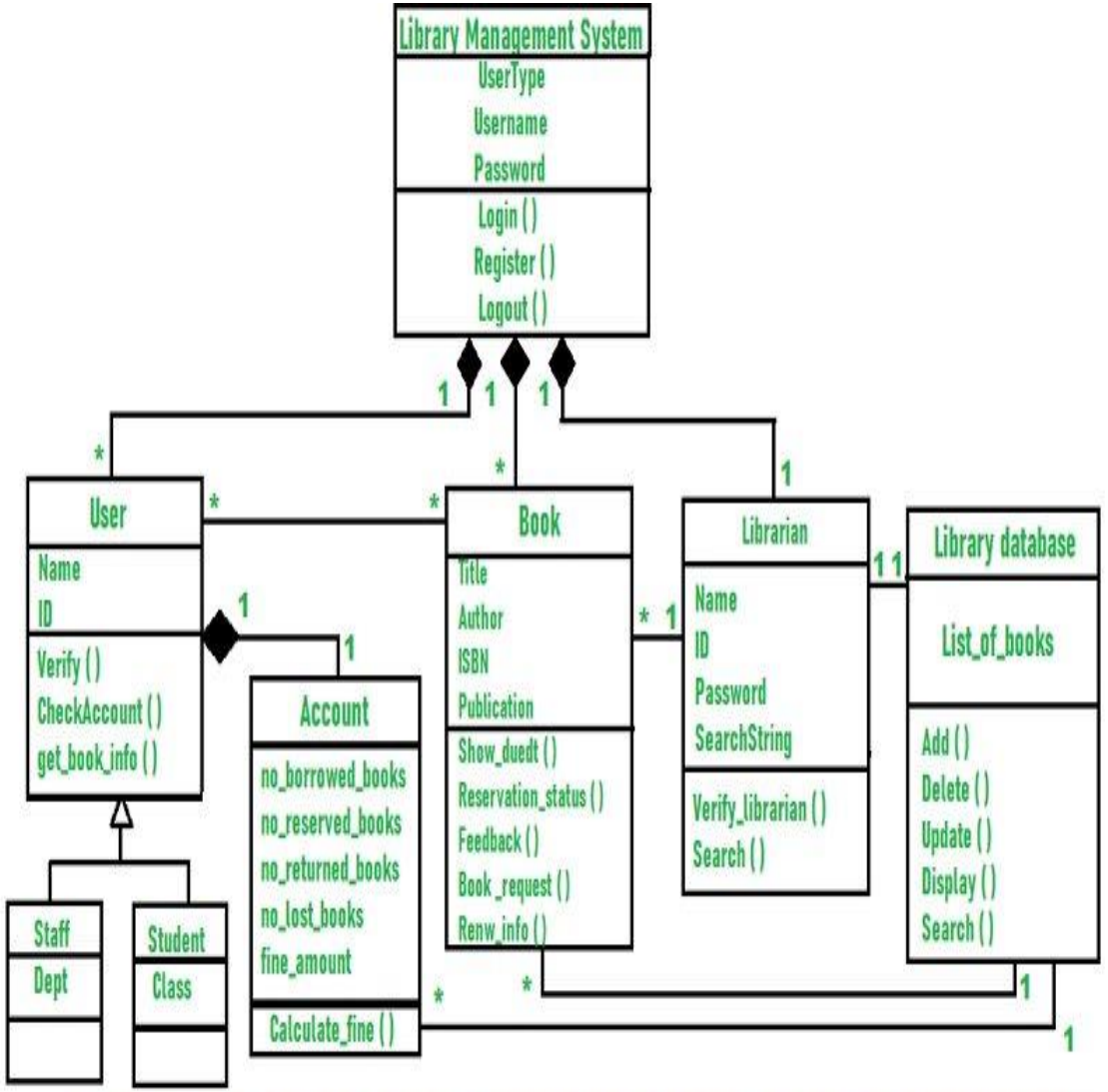
MERISE


UML Diagrams
Behaviour Diagrams
Structure Diagrams
Activity
Sequence
State Machine
Communication
Use Case
Class
Database
Component
Deployment


Online Shopping System
View items
<<include>>
Make purchase
<<include>>
Complete checkout
Log in
Customer
<<service>>
Authentication
Identity Provider
Credit Payment Service
PayPal


Author
Reviewer
Approver
Owner
Create
Update
Document [Draft]
Change Request
Review Draft
Review Effective
[change needed]
[obsolete]
Document [Reviewed]
[not approved]
Approve
[approved]
Document [Effective]
Archive
Document [Archived]
© uml-diagrams.org





CLASS DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM

Documentation:

- CAHIER DES CHARGES :
  - Réaliser par l'AMOA (Assistant à Maitrise d'Ouvrage)
  - A une valeur contractuelle

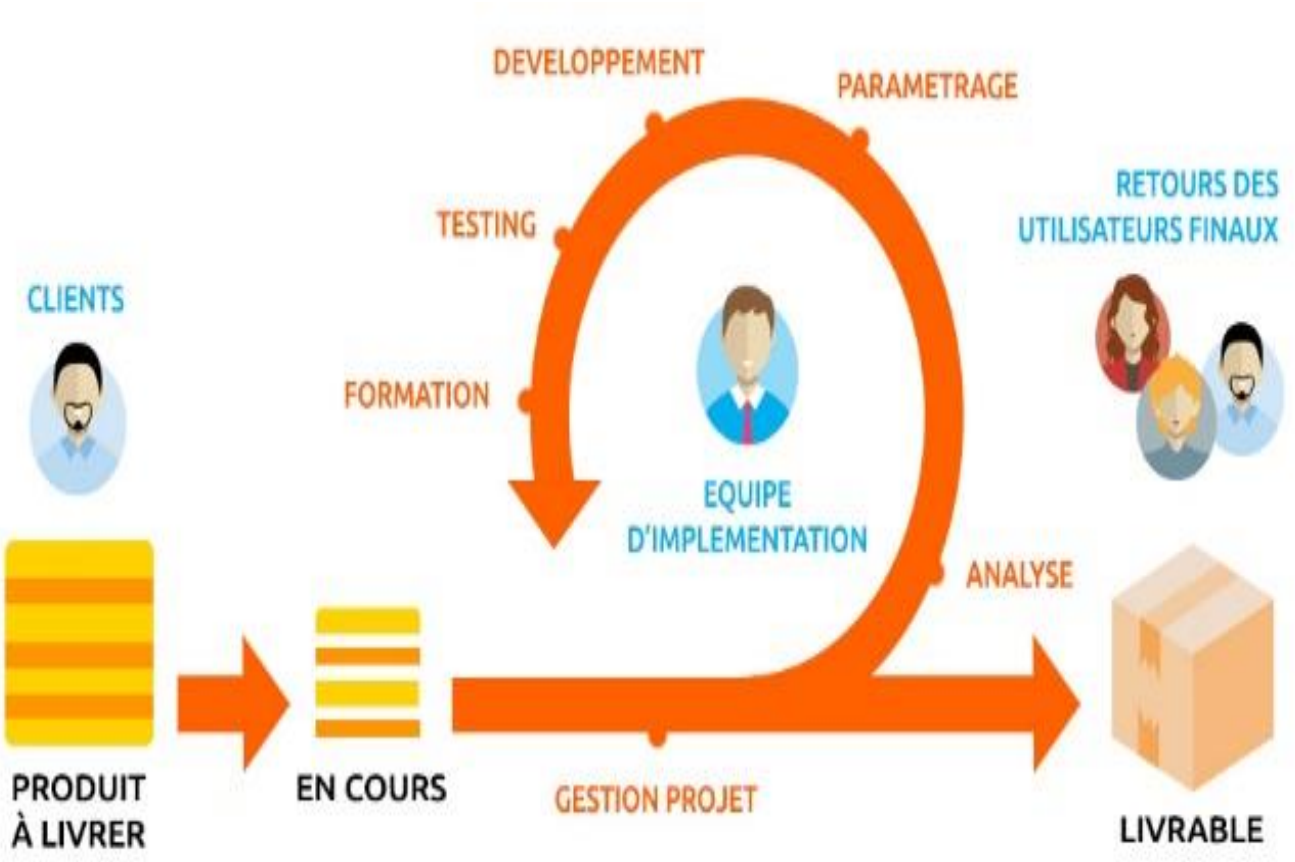
- SPECIFICATIONS :
  - Fonctionnelles : Décrit les Fonctionnalités attendues avec un langage Métier
  - Techniques : Idem avec un langage Informatique

2. La Gestion « New School »

Les Premiers cycles ayant leur limites, on a repenser le système.

La gestion est maintenant pérenne et adaptative

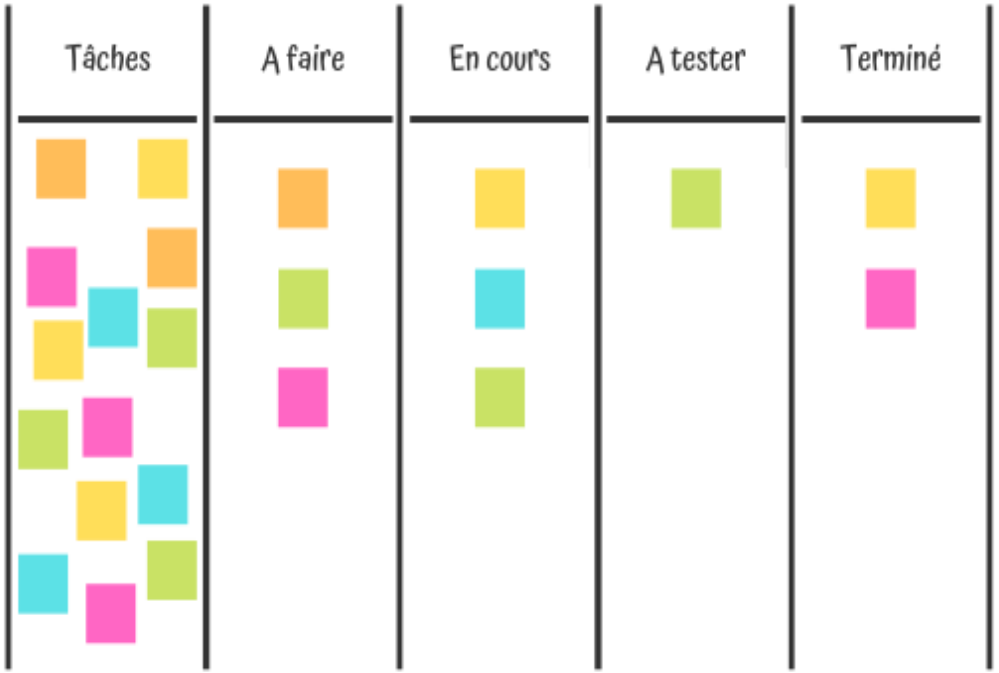
AGILITY : La Méthode SCRUM

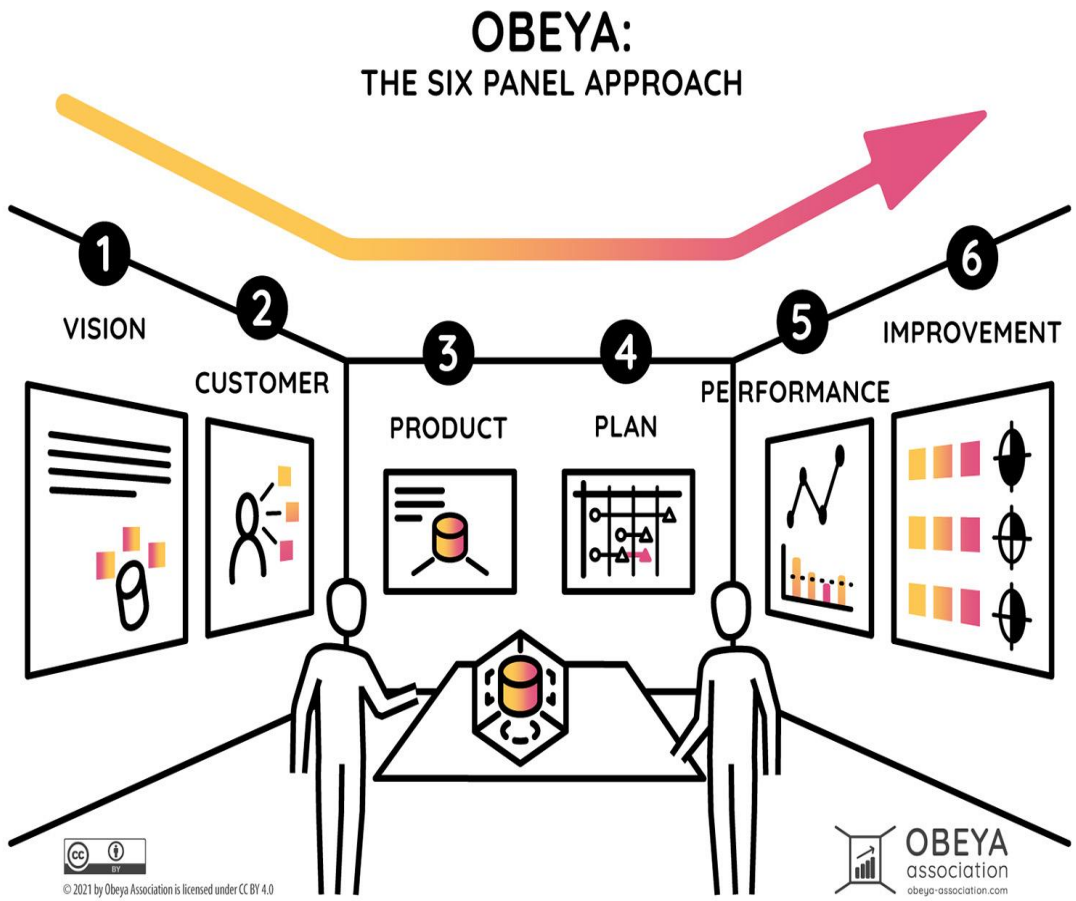


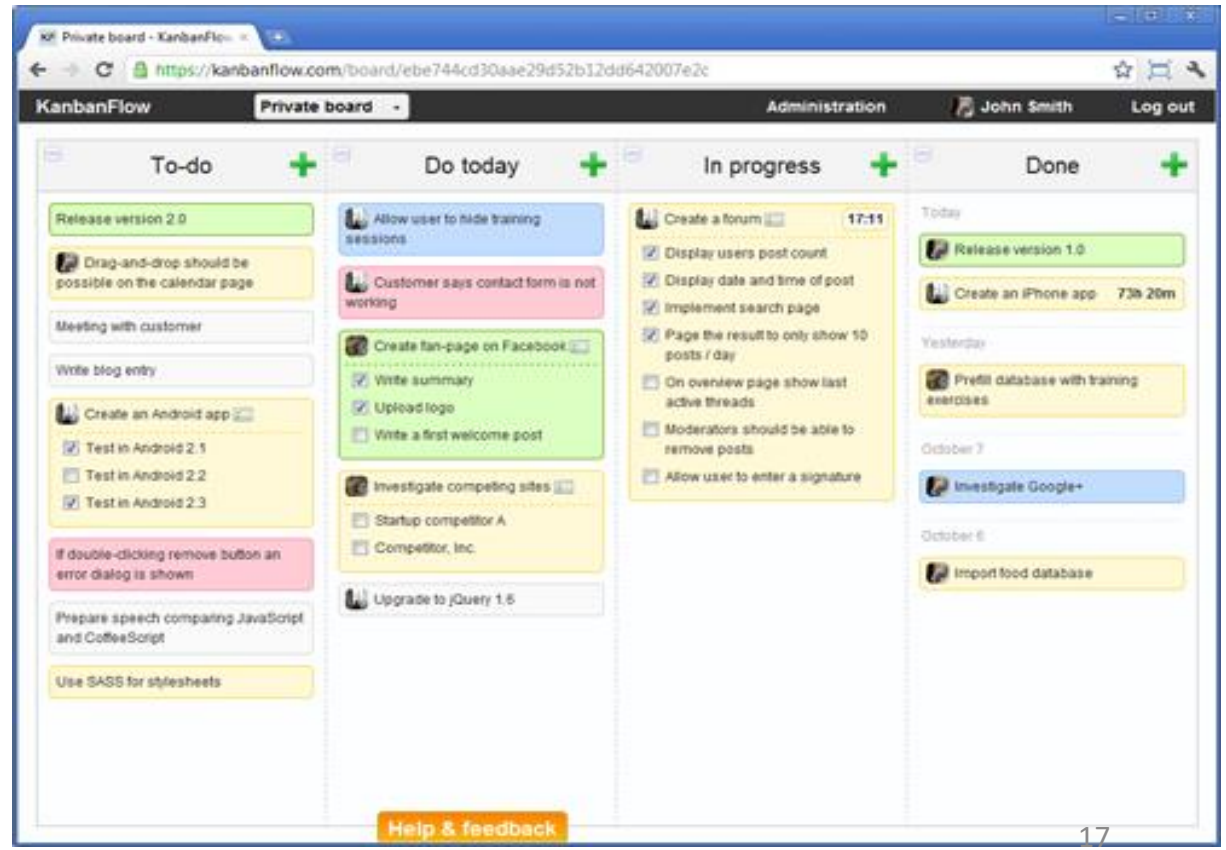
- Itérations Incrémentales
- Découpage du travail en Sprints
- Inclusion des Tests fonctionnels pour les Développeurs
- Rituel des Réunions (Hyper communication) (Daily, Revue de Sprint…)
- Inclusion d'un Product Owner
- Notion de Scrum Master
- Feed-Back des clients finaux

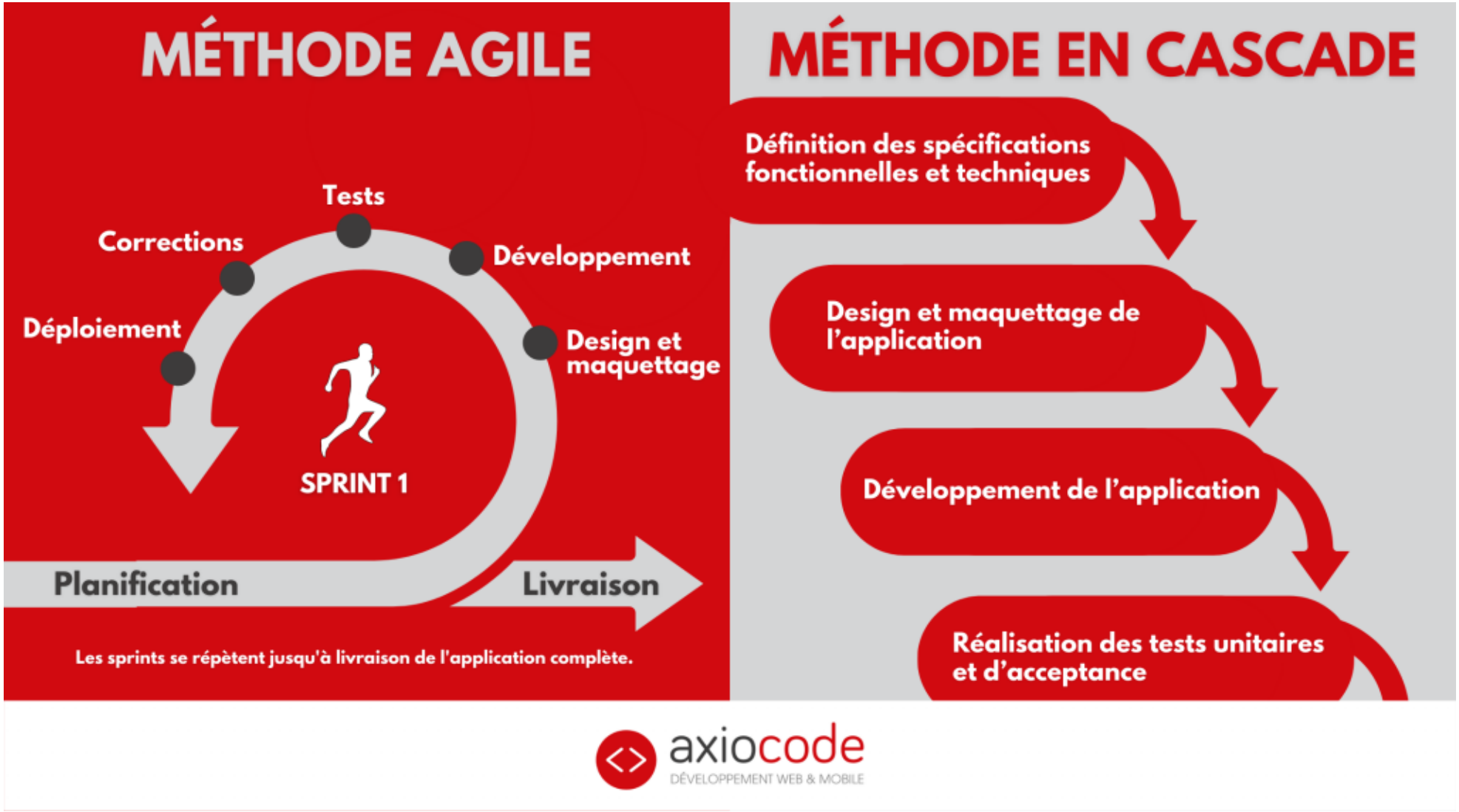
AGILITY : Les Outils KANBAN

- Représentation d'un Sprint
- Responsabilité de chacun et visibilité globale
- Prise en considération des nouvelles idées (Back Log ou Bac à Sable)
- Animer les réunions Scrum



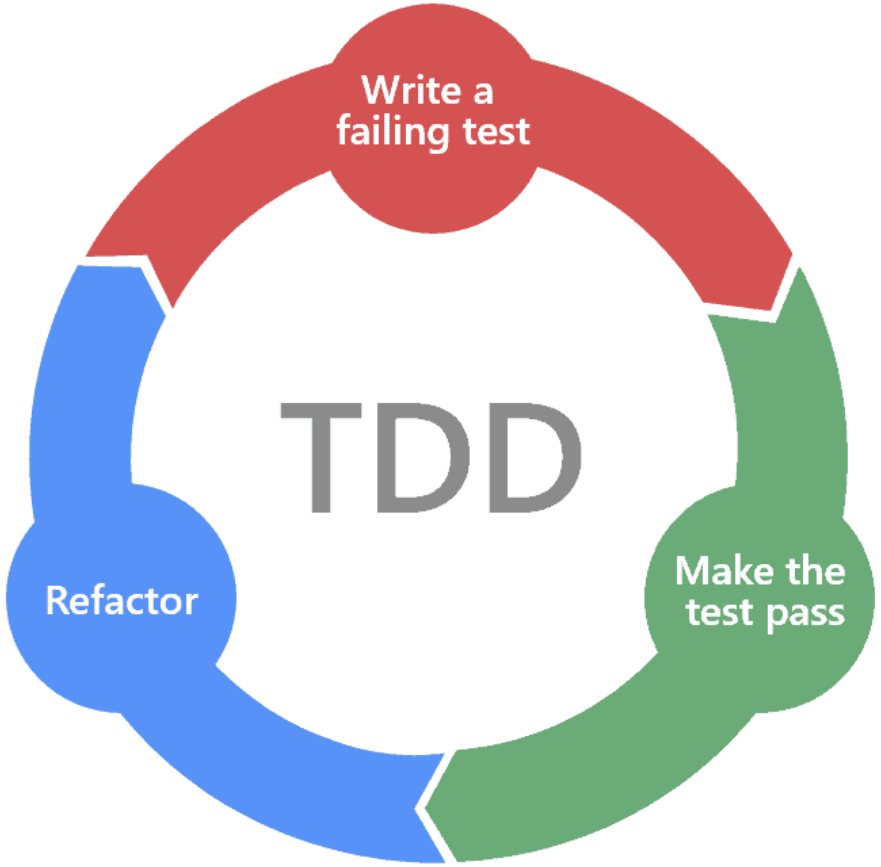

OBEYA:
THE SIX PANEL APPROACH
1
VISION
2
CUSTOMER
3
PRODUCT
4
PLAN
5
PERFORMANCE
6
IMPROVEMENT
© 2021 by Obeya Association is licensed under CC BY 4.0
OBEYA
association
obeya-association.com

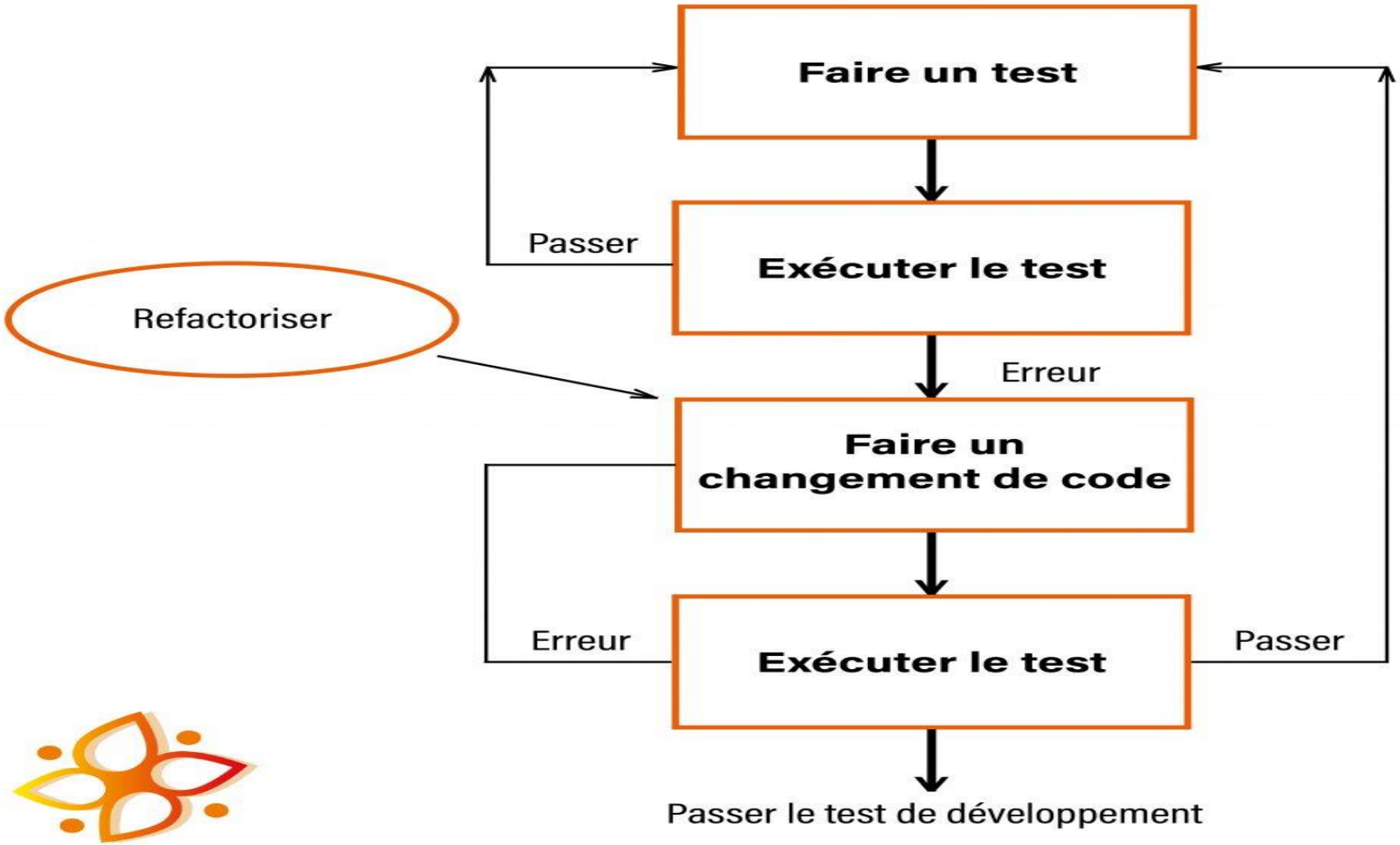

KanbanFlow
Private board
Administration
John Smith
Log out
To-do
Do today
In progress
Done
Release version 2.0
Drag-and-drop should be possible on the calendar page
Meeting with customer
Write blog entry
Create an Android app
Test in Android 2.1
Test in Android 2.2
Test in Android 2.3
If double-clicking remove button an error dialog is shown
Prepare speech comparing JavaScript and CoffeeScript
Use SASS for stylesheets
Allow user to hide training sessions
Customer says contact form is not working
Create fan-page on Facebook
Write summary
Upload logo
Write a first welcome post
Investigate competing sites
Startup competitor A
Competitor, Inc.
Upgrade to jQuery 1.6
Create a forum
17:11
Display users post count
Display date and time of post
Implement search page
Page the result to only show 10 posts / day
On overview page show last active threads
Moderators should be able to remove posts
Allow user to enter a signature
Release version 1.0
Create an iPhone app
73h 20m
Prefill database with training exercises
Investigate Google+
Import food database
Help & feedback



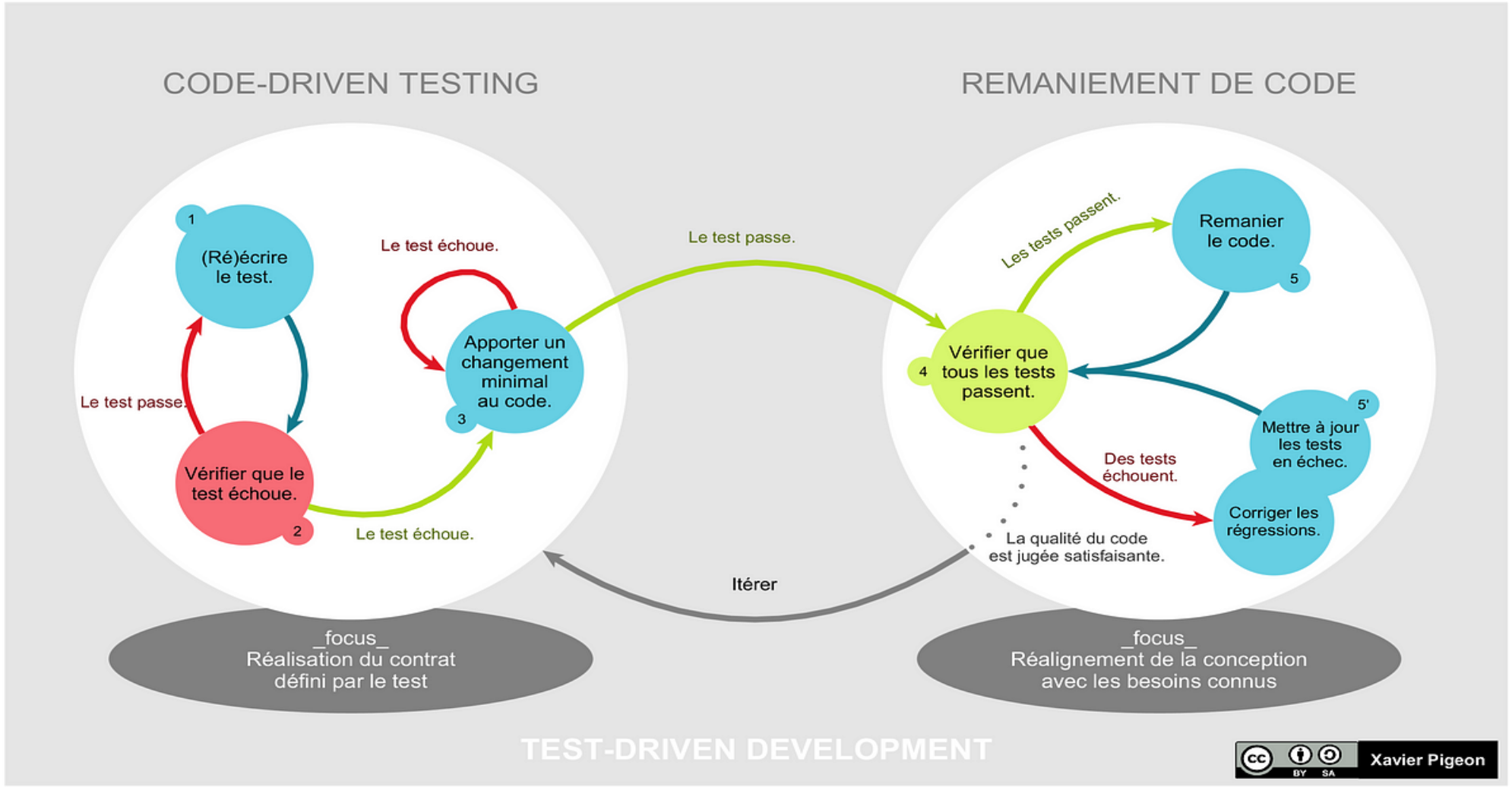
Comparaison

TDD, BDD, DDD




Faire un test
Passer
Exécuter le test
Refactoriser
Erreur
Faire un changement de code
Erreur
Exécuter le test
Passer
Passer le test de développement

Simple TDD


CODE-DRIVEN TESTING
REMANIEMENT DE CODE
1
(Ré)écrire le test.
Le test échoue.
Le test passe.
Apporter un changement minimal au code.
3
Le test passe.
Vérifier que le test échoue.
2
Le test échoue.
Les tests passent.
Remanier le code.
5
4
Vérifier que tous les tests passent.
5'
Mettre à jour les tests en échec.
Des tests échouent.
Corriger les régressions.
La qualité du code est jugée satisfaisante.
Itérer
focus
Réalisation du contrat défini par le test
focus
Réalignement de la conception avec les besoins connus
TEST-DRIVEN DEVELOPMENT
Xavier Pigeon

Détail TDD

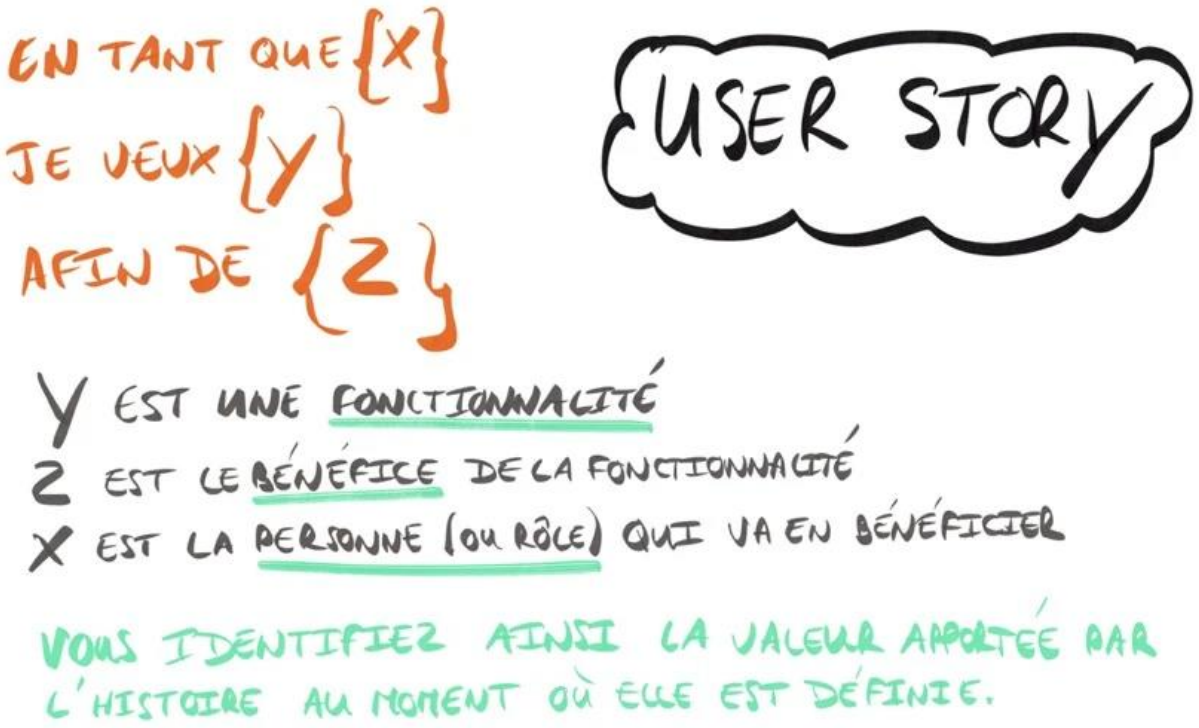
BDD (Behaviour Driven Developpement)

- Se base sur des User Stories pour élaborer les nouvelles fonctionnalités

- S’incére dans une logique Agile

  La user Story permettant un découpage en tâches pour les développeurs

Patron de User Story :



Patron avec scénario :

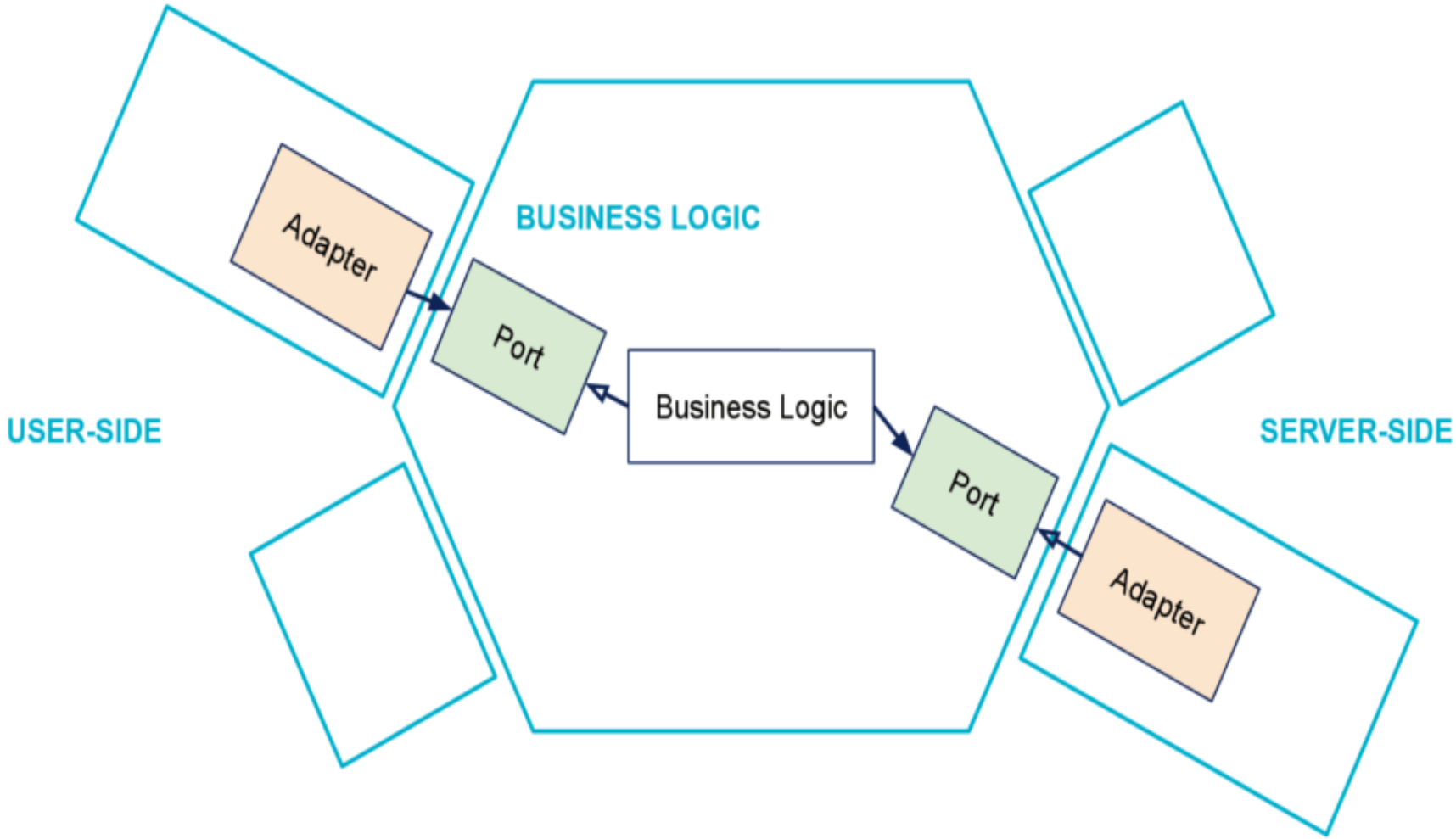
Title (one line describing the story)
Narrative:
    As a [role]
    I want [feature]
    So that [benefit]
Acceptance Criteria: (presented as Scenarios)
Scenario 1: Title
    Given [context]
    And [some more context] ...
    When [event]
    Then [outcome]
    And [another outcome] ...
Scenario 2: ...

DDD (Domain Driven Design)

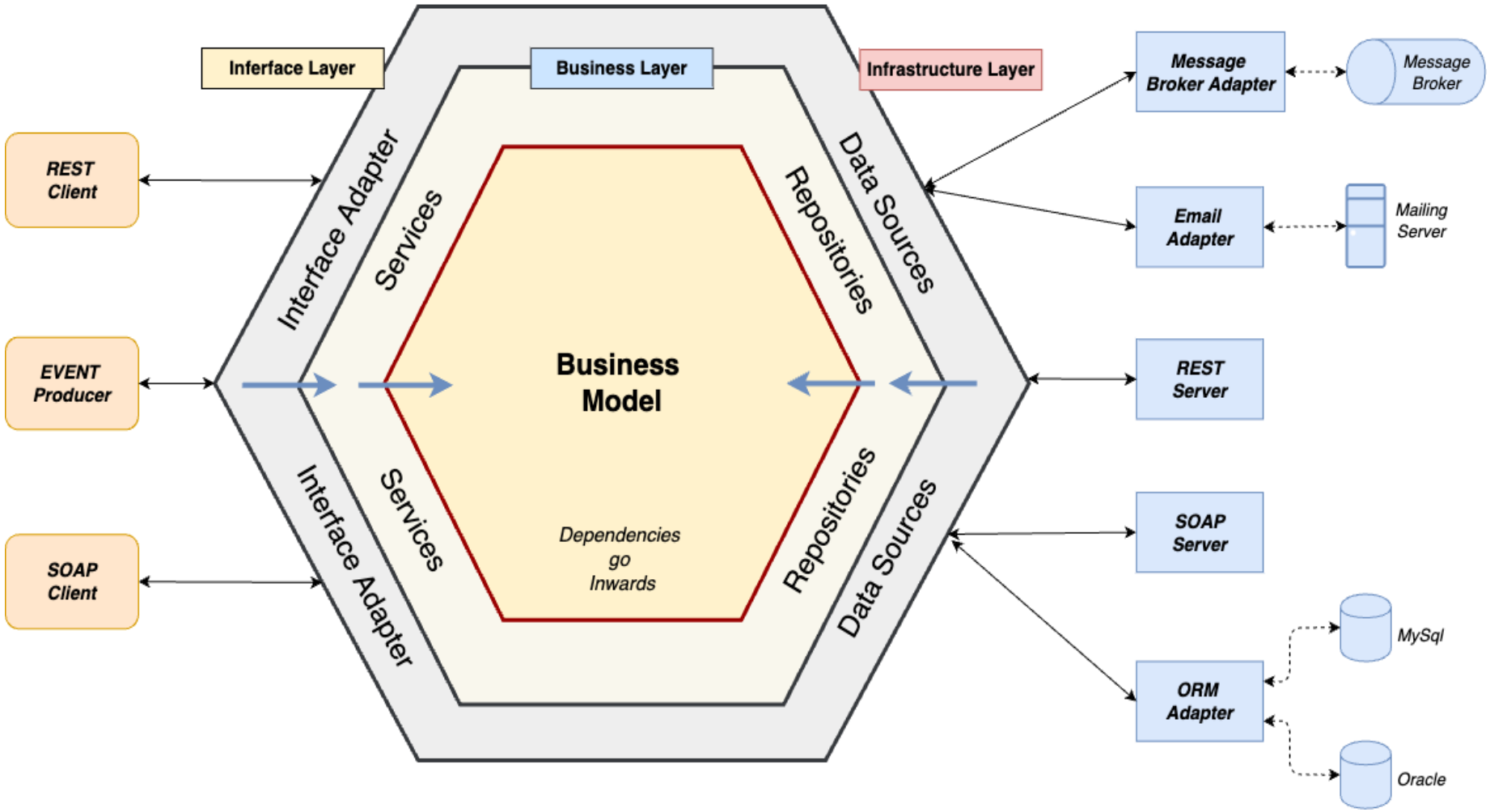
- Se base sur la modélisation (UML ou autres)
- Se base sur le Domain (Métier) et la logique associée
- Se base sur les bons principes d'architecture logiciel comme les Design Patterns et l'architecture Hexagonale.

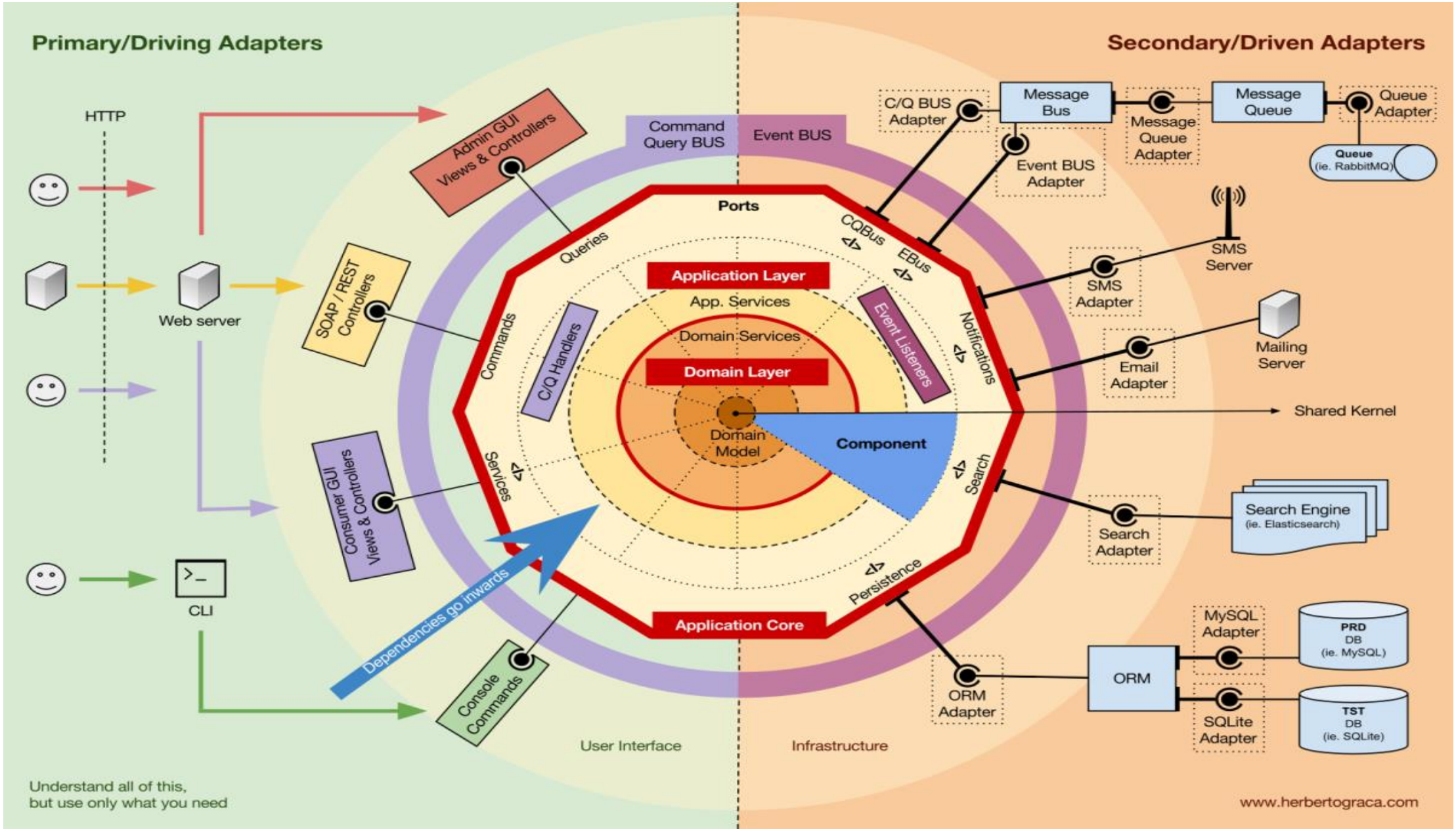
  Chaque domaine doit dépendre d'un noyau, un noyau ne dépend d'aucun domaine


Adaptateurs
IU
Adaptateur de tests
Adaptateur journalisation
Ligne de commande
Adaptateur SMS
Adaptateur courriel
Adaptateur SGBD
Adaptateur fichier
Application (noyau)
Notification
Persistance
Administration


BUSINESS LOGIC
Adapter
Port
Business Logic
Port
Adapter
USER-SIDE
SERVER-SIDE

Hexagonal Architecture




Primary/Driving Adapters
Secondary/Driven Adapters
HTTP
Web server
CLI
Admin GUI
Views & Controllers
SOAP / REST
Controllers
Consumer GUI
Views & Controllers
Console
Commands
Command
Query BUS
Event BUS
Ports
Queries
Commands
Application Layer
App. Services
Domain Services
Domain Layer
Domain
Model
C/Q Handlers
Event Listeners
CQBus
EBus
Notifications
Services
Search
Persistence
Component
Application Core
Shared Kernel
Dependencies go inwards
C/Q BUS
Adapter
Message
Bus
Event BUS
Adapter
Message
Queue
Adapter
Message
Queue
Queue
Adapter
Queue
(ie. RabbitMQ)
SMS
Server
SMS
Adapter
Mailing
Server
Email
Adapter
Search Engine
(ie. Elasticsearch)
Search
Adapter
MySQL
Adapter
PRD
DB
(ie. MySQL)
ORM
ORM
Adapter
SQLite
Adapter
TST
DB
(ie. SQLite)
User Interface
Infrastructure
Understand all of this,
but use only what you need
www.herbertograca.com

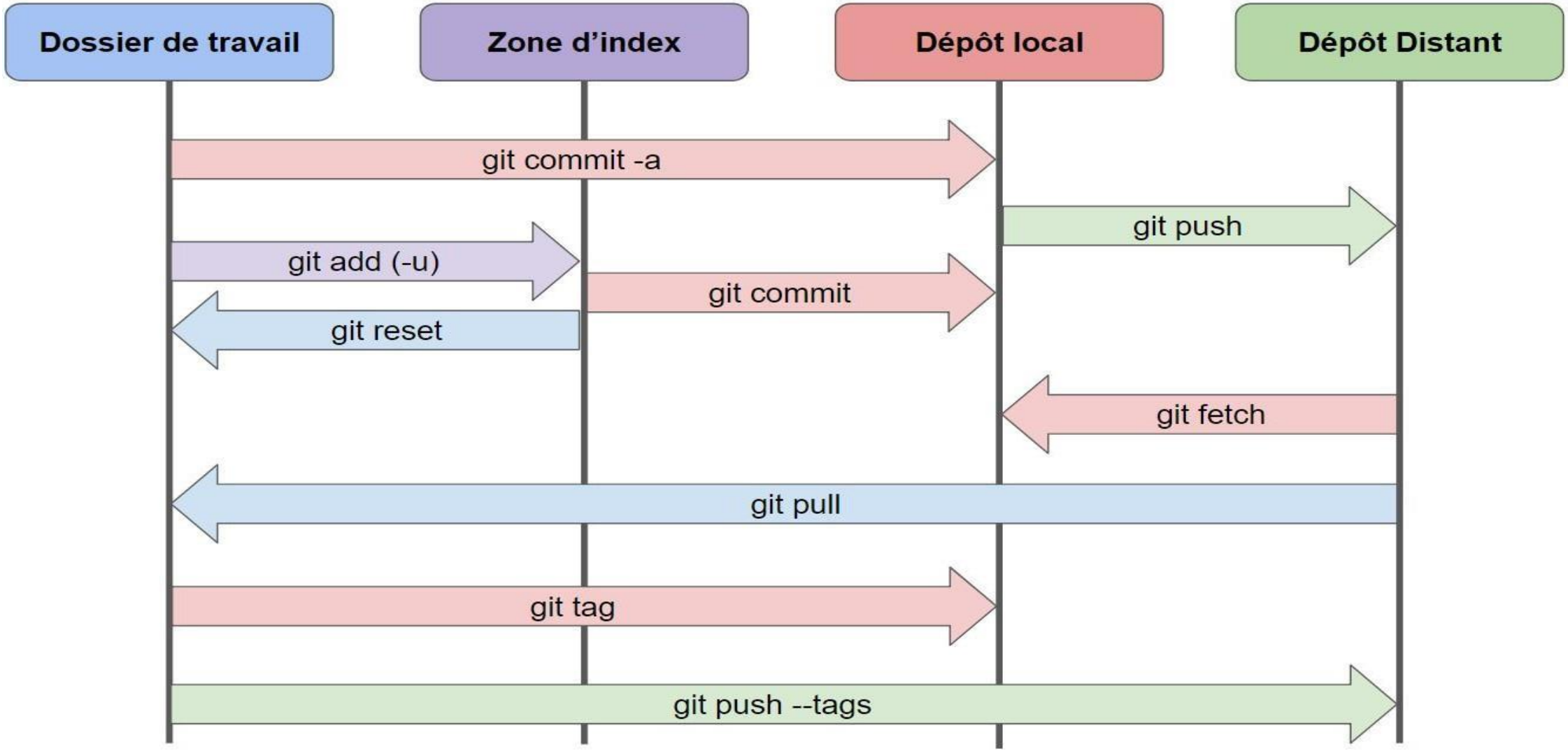
PARTIE 2 : Focus Versioning

1. Installation du logiciel GIT

Linux: sudo apt-install git-all

Mac: https://git-scm.com/download/mac

Windows: https://git-scm.com/download/win


Dossier de travail
Zone d'index
Dépôt local
Dépôt Distant
git commit -a
git push
git add (-u)
git commit
git reset
git fetch
git pull
git tag
git push --tags

2. Utilisation Locale

- **git init [nom_dépôt] :**

Crée un dépôt local vide dans le dossier courant

ou en créer un avec le nom spécifié.

- **git config --global user.name « [nom] » :**

Définit le nom de l'utilisateur

- **git config --global user.email [email] :**

Définit l'email de l'utilisateur

N.B :

On peut ajouter un fichier .gitignore dans lequel on spécifie à GIT quels fichiers ignorer.

On peut en mettre un à la racine du projet pour tout le projet ou en placer un dans chaque dossier.

- **git remote add [url] :**

Ajoute comme remote le dépôt pointé par l’url.

(En complément de git init pour lier un projet local à un dépôt distant)

- **git clone [url] :**

Duplique en local le dépôt git pointé par l’url.

(Lorsqu’on rejoint un projet en cours)

Commandes de Base :

- **git add –a :**

Ajoute toute les modifications au STAGE

- **git commit -m « [message] » :**

Crée un commit avec un message

- **git log [-n] :**

Montre l'historique des commits pour la branche courante. -n pour limiter aux n derniers commits.

- **git tag [nom_tag] :**

Créer un tag sur le commit courant

- **git checkout [fichier] :**

Supprime les modification courantes du fichier. (Attention le comportement de checkout dépend des arguments que vous lui passer)

Commandes de Correction :

- **git commit --amend :**

Ajoute au dernier commit le contenue de la zone d’index et change le message de commit.

- **git revert [commit] :**

Créer un nouveau commit qui est l'opposé du commit spécifié afin d’annuler ses effets.

- **git reset [commit] :**

Annule tous les commits après `[commit]`, en conservant les modifications localement.

- **git reset --hard [commit] :**

Supprime tout l'historique et les modifications effectuées après le commit spécifié.

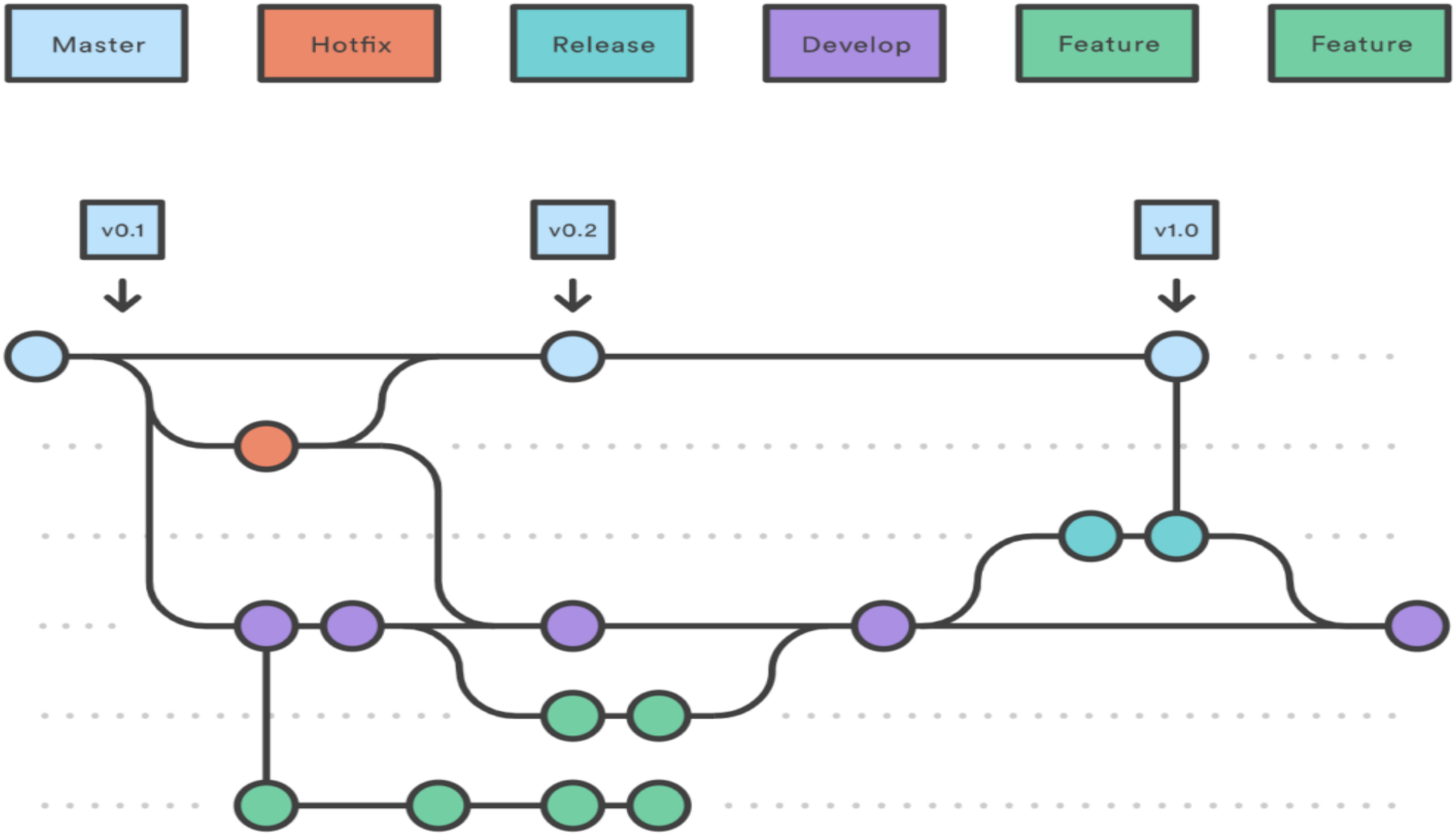
Mettre de Côté :

- **git stash save « [message] » :**

Enregistre toutes les modifications courante dans une pile temporairement.

- **git stash list :**

Liste toutes modifications mises de côté.


Master
Hotfix
Release
Develop
Feature
Feature
v0.1
v0.2
v1.0

Travail sur des Branches :

- **git branch [nom_branche] :**

Crée une nouvelle branche, du nom de nom_branche
(option –d pour la supprimer)

- **git checkout -b [nom_branche] :**

Change de branche et ce placer sur le dernier commit de celle-ci,
-b pour en plus créer la branche

- **git branch -r –a :**

Liste toutes les branches locales dans le dépôt
courant. -r pour les branche distantes. -a pour toutes les locales

- **git merge [autre_branche] :**

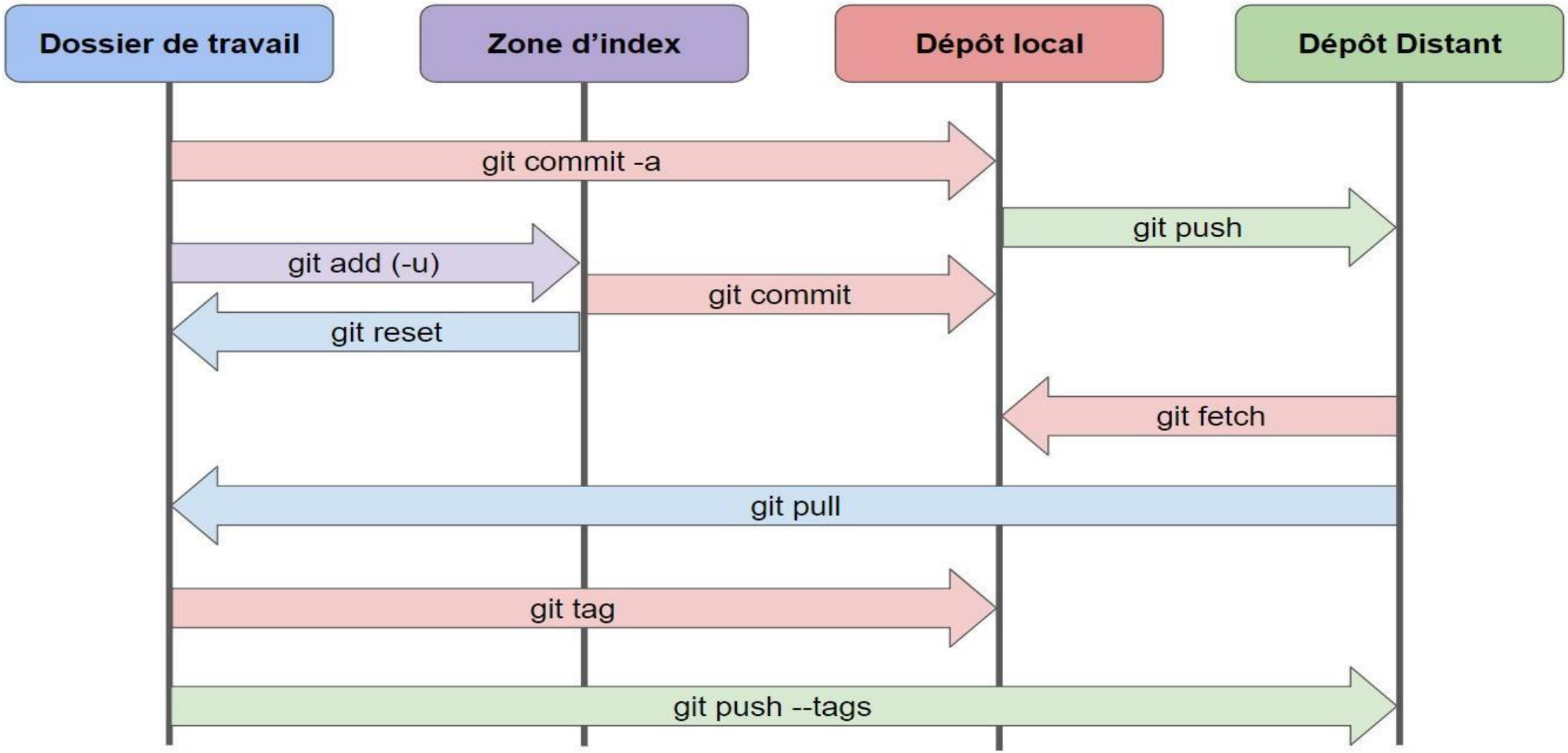
Combine dans la branche courante l'historique de la branche spécifiée via un commit de merge

- **git rebase [autre_branche] :**

Déplace les commits de la branche courante sur la branche spécifiée

- **git cherry-pick [-x] [commit] :**

Applique un commit à l’espace de travail. -x permet d’ajouter le message « cherry-picked from commit [sha1_commit] »


Dossier de travail
Zone d'index
Dépôt local
Dépôt Distant
git commit -a
git push
git add (-u)
git commit
git reset
git fetch
git pull
git tag
git push --tags

3. Utilisation Distante

- **git fetch [remote]:**

Met à jour les informations locales concernant le serveur distant

- **git pull [remote] [branch] :**

Met à jour son dépôt local avec les commits présents sur le serveur distant

- **git push [remote] [branch] :**

Envoie les commits en local sur le serveur distant

Rendez-vous sur https://github.com/MoradDev
pour une compréhension de la web-application

PARTIE 3 : Focus TDD

1. Utilisation de La Bibliothèque unittest de PYTHON

- Dans le fichier de tests, importer unittest

```
import unittest
```

- Instancier l'objet à tester

```
instance=MaClass()
```

- A l'intérieur de la class de test, créer des fonctions qui commencent obligatoirement par « test_ » et qui prennent en paramètre self

```
def test_une_fonctionnalite(self):
```

- Créer un test dedans en utilisant les assertions

```
self.assertEqual(self.instance.aire_cercle(1), math.pi)
```

2. Les Méthodes de unittest

- La méthode setUp() permet d'y mettre ce qu'on veut faire avant chaque test

```
def setUp(self):
    print("Nouveau Test")
    self.instance=MaClass("premiere_class")
```

- La méthode tearDown() elle fait le contraire, c'est le traitement après chaque test

```
def tearDown(self):
    print("Fin de ce test")
```

- La méthode main() permet de lancer les tests de la class en lançant simplement le fichier de test. Pour cela on l'a mettra dans un if qui test si le fichier est directement lancer

```
if __name__ == "__main__":
    unittest.main()
```

- La méthode subTest() permet de créer plusieurs tests dans une même méthode de test, grâce à l'expression « with self.subTest(self) : »

```
def test_aire_cercle(self):
    '''tests de aire_cercle()'''
    with self.subTest(self):
        self.assertEqual(self.instance.aire_cercle(0), 0)

    with self.subTest(self):
        self.assertEqual(self.instance.aire_cercle(1), math.pi)
```

3. Les assertions

Ouvrez votre IDE

4. Lecture de Tests

```
PS C:\Users\MORADK\Desktop\Projets_Python\TDD_Python> & C:/Users/MORADK/AppData/Local/Prog
rams/Python/Python311/python.exe c:/Users/MORADK/Desktop/Projets_Python/TDD_Python/test_ma
_class.py
Nouveau Test
Fin de ce test
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
```

```
PS C:\Users\MORADK\Desktop\Projets_Python\TDD_Python> & C:/Users/MORADK/AppData/Local/Prog
rams/Python/Python311/python.exe c:/Users/MORADK/Desktop/Projets_Python/TDD_Python/test_ma
_class.py
Nouveau Test
FFFin de ce test

======================================================================
FAIL: test_aire_cercle (__main__.MonTest.test_aire_cercle) [test_aire_cercle (__main__.Mon
Test.test_aire_cercle)]
tests de aire_cercle()
----------------------------------------------------------------------
Traceback (most recent call last):
  File "c:\Users\MORADK\Desktop\Projets_Python\TDD_Python\test_ma_class.py", line 15, in t
est_aire_cercle
    self.assertEqual(self.instance.aire_cercle(0), 0)
AssertionError: 1.0 != 0

======================================================================
======================================================================
FAIL: test_aire_cercle (__main__.MonTest.test_aire_cercle) [test_aire_cercle (__main__.Mon
Test.test_aire_cercle)]
tests de aire_cercle()
----------------------------------------------------------------------
Traceback (most recent call last):
  File "c:\Users\MORADK\Desktop\Projets_Python\TDD_Python\test_ma_class.py", line 18, in t
est_aire_cercle
    self.assertEqual(self.instance.aire_cercle(1), math.pi)
AssertionError: 4.141592653589793 != 3.141592653589793

----------------------------------------------------------------------
Ran 1 test in 0.002s

FAILED (failures=2)
```

```python
class MonTest(unittest.TestCase):

    def setUp(self):
        print("Nouveau Test")
        self.instance=MaClass("premiere_class")

    def test_aire_cercle(self):
        '''tests de aire_cercle()'''
        with self.subTest(self):
            self.assertEqual(self.instance.aire_cercle(0), 0)

        with self.subTest(self):
            self.assertEqual(self.instance.aire_cercle(1), math.pi)

    def tearDown(self):
        print("Fin de ce test")


if __name__ == "__main__":
    unittest.main()
```