

Soutenance de Génie logiciel et de Projet de Développement



Antoine Vidal-Mazuy
Jean Philippe Carlens
Yann Brault
Luca Orengo
Antoine Cousson



Maven



sonarqube



Antoine Vidal-Mazuy : Serveur, Stratégie

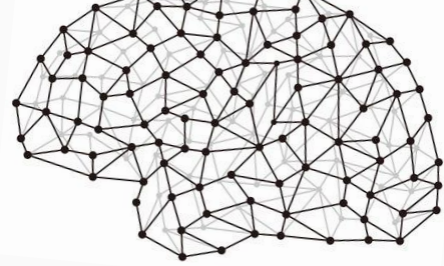
Jean Philippe Carlens : IO , Choix IA

Yann Brault : Cartes, Effets

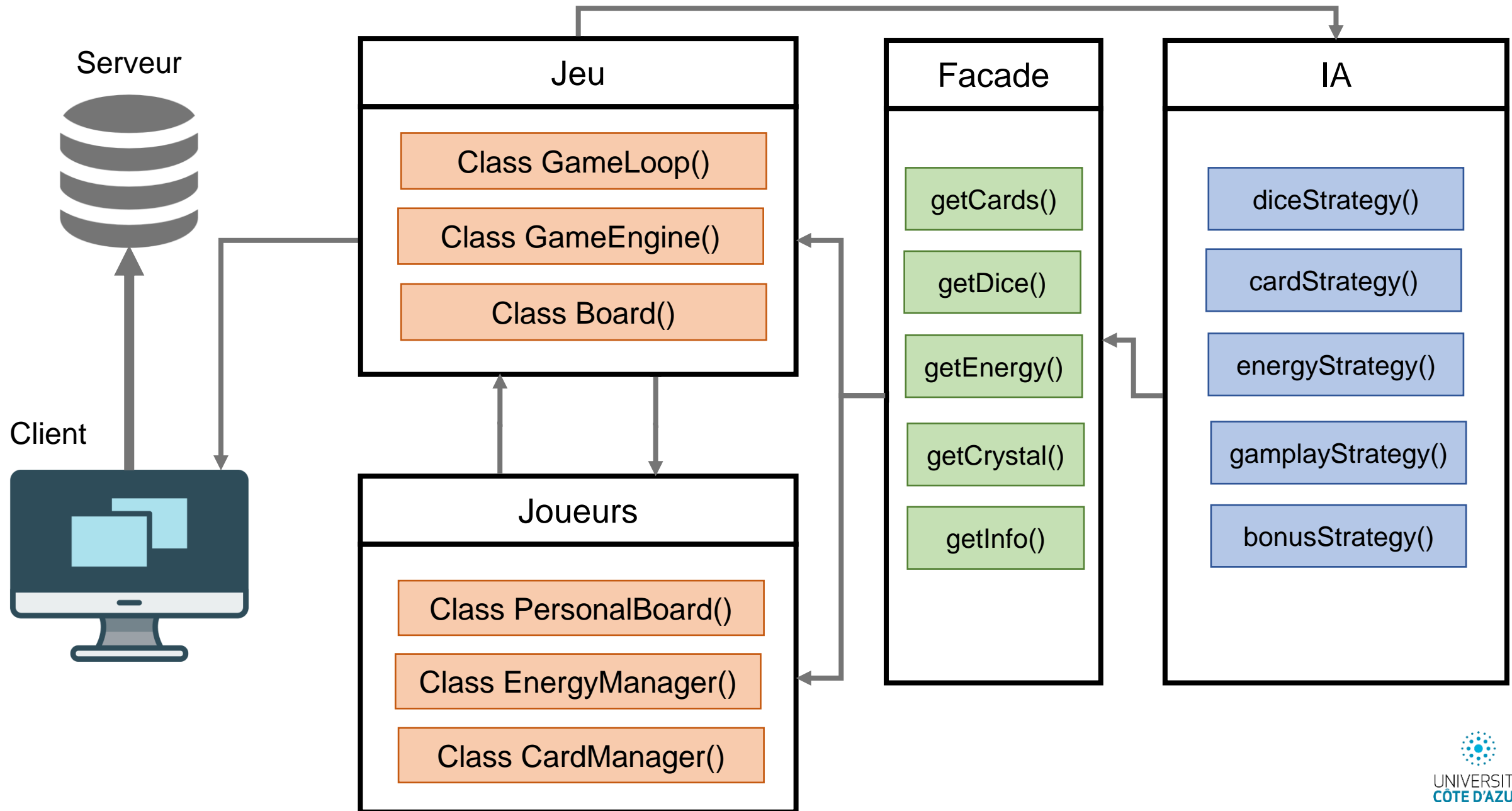
Luca Orenco : IA Initiales

Antoine Cousson : Game Observer, Façade

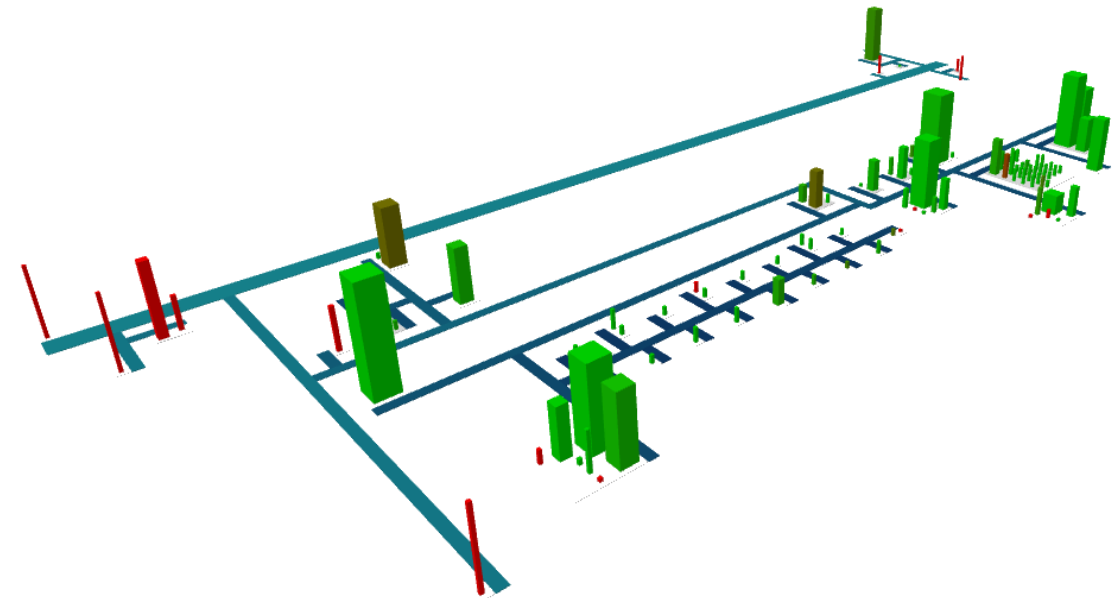
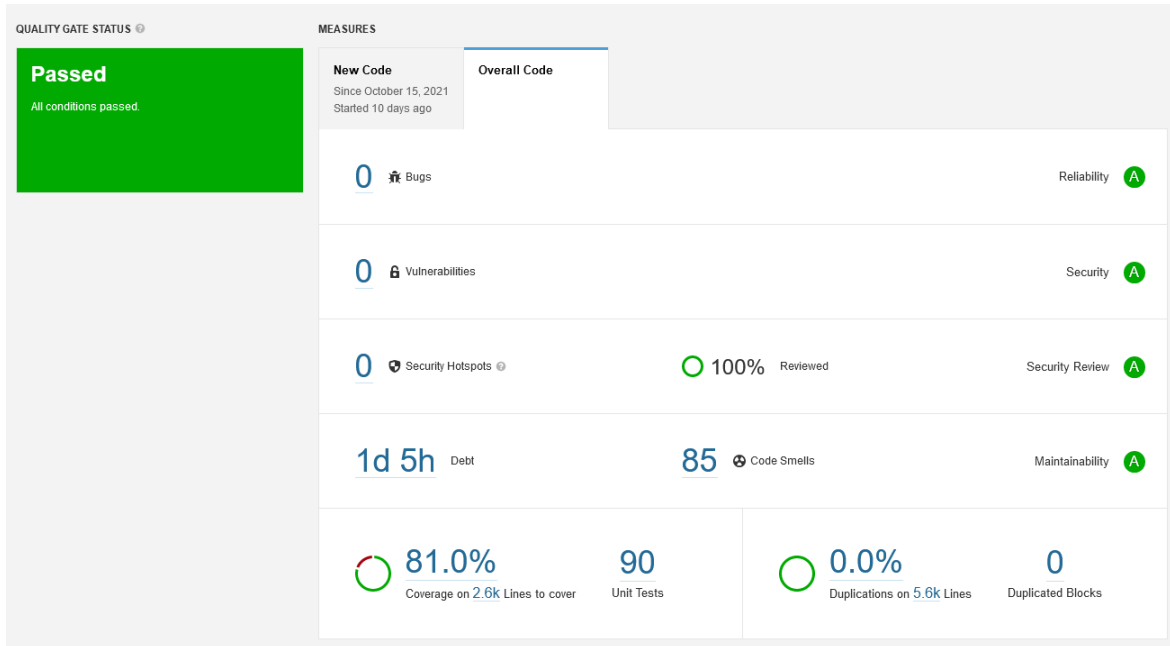
```
class PlayerTurnActionsProcessor {  
  
    private final PlayerTurnManager playerTurnManager;  
    private final Player player;  
    private final PersonalBoard personalBoard;  
    private final GameEngine engine;  
  
    PlayerTurnActionsProcessor(PlayerTurnManager playerTurnManager) {  
        this.playerTurnManager = playerTurnManager;  
        this.player = playerTurnManager.getPlayer();  
        this.personalBoard = playerTurnManager.getPersonalBoard();  
        this.engine = playerTurnManager.getEngine();  
    }  
  
    void processActivation() {  
        ArrayList<Card> cardToActivate = new ArrayList<>();  
  
        for (Card card : personalBoard.getCardManager().getInvokeDeck().getCards()) {  
            if (card.getEffect().getIsActivationEffect() && card.getEffect().canActivate(engine, player)) {  
                cardToActivate.add(card);  
            }  
        }  
  
        if (!cardToActivate.isEmpty()) {  
            player.getFacadeIA().setCardToChooseF(cardToActivate);  
            Card c = player.chooseCardToActivate();  
            playerTurnManager.getDescription().append(String.format("%n\t→ chooses to activate %s : ", c.getName()));  
            c.getEffect().applyEffect(engine, playerTurnManager.getPlayer());  
            engine.getStatsManager().addActivatedCards(player, amount: 1);  
        }  
    }  
}
```



Fonctionnalités	Confiance de la Réalisation	Confiance du Code
<ul style="list-style-type: none"> ✓ 30 premières cartes ✓ Effets ✓ Types d'effets ✓ Types de cartes ✓ Dés ✓ IA ✓ Client/serveur ✓ Statistiques ✓ Loggers ✗ 20 dernières cartes 	<ul style="list-style-type: none"> ✓ Tests tout au long du projet ✓ Organisation Multi-Module ✓ Utilisation de SonarQube ✓ Prise en compte des métriques ✓ Patrons de Conception ✓ Découplage des responsabilités ✗ Découplage partiel des Classes Managers ✗ Complexité cyclomatique des Managers 	<ul style="list-style-type: none"> ✓ 1 ou plusieurs Gherkin par effets ✓ Effets en général ✗ Combinaison d'effets complexes ✗ Actualisation des choix des IA



Patrons / Antipatrons	Implémentation
Factory	IA / Effect
Singleton	Dés / Cartes(Load via JSON)
Strategy	Choix IA
GameLoop	Cœur du Jeu
Observer	IA observe le Jeu
Façade	Isolement des IA
Poltegeist Class	PlayerTurnManager => PlayerTurnActionProcessor



Footprint: Cyclomatic Complexity

Height: Lines of Code

Color: Coverage

UNIVERSITÉ
CÔTE D'AZUR

- ✓ Bon découpage des tâches au début et à la fin du projet
- ✓ Projet Incrémentable
- ✓ Projet malléable
- ✓ Respect du MVP à chaque livraisons
- ✓ Bonne couverture des tests
- ✓ Dette technique satisfaisante
- ✓ Peu de codes smells
- ✓ Utilisation de GitHub/Kanban
- ✓ Respect des règles de Seasons
- ✓ Moteur de jeu caché aux IA



- × Moins bon découpage des Users Stories vers le milieu du projet
- × Complexité cyclomatique élevée des Managers
- × 1 Carte, 1 effet
- × Couverture de test sonarQube douteuse
- × Pas assez de tests pointus
- × Pertinence des tests



Découvertes à conserver	À améliorer	Habitudes à supprimer
<ul style="list-style-type: none">• Gherkin• SonarQube• Git• User Story• MVP• 1 commit par tâches• Projets Multi Modules• Maven	<ul style="list-style-type: none">• Découpage tâches• Gestion livraisons• Javadoc• Tests Unitaires• Gherkin reliés aux Users Stories• Mockito• Gestion des dépendances	<ul style="list-style-type: none">• Crunch• Commits trop lourd• Plusieurs tâches par commit• Mauvaises descriptions des commits

Remerciements



Antoine Vidal-Mazuy
Jean Philippe Carlens
Yann Brault
Luca Orengo
Antoine Cousson



Maven[™]



sonarqube

