

Conception et GRASP

8INF327 – Modélisation et
développement objets

1

Journée(s) de modélisation

- Suggestion de fonctionnement
 - Séparer l'équipe en plusieurs petits groupes
 - Définir un temps de travail assez court, par exemple, une demi-journée
 - Débuter les premières itérations en mode esquisse, éviter les logiciels de modélisation
 - Créer les modèles **pour comprendre et argumenter et non pas pour documenter**
 - Ceux-ci viennent plus tard pour la génération de code? Et encore...
- Ultra important
 - Ne **JAMAIS** viser un modèle complet
 - Les programmeurs doivent programmer. C'est la seule vraie méthode de mettre le modèle à l'épreuve (**80%/20%**)

6

Conception pilotée par les responsabilités

Chaque objet dans un système a des responsabilités

- Les responsabilités de **faire** d'un objet peuvent être :
 - Effectuer une **action** particulière
 - Créer un autre objet
 - Charger un traitement
 - **Déclencher** une action dans un autre objet
 - Exécuter une règle
 - **Contrôler** et **coordonner** les activités d'autres objets
 - La boucle de jeu, les entrées et sorties d'une « donnée »

7

Conception pilotée par les responsabilités

- Les responsabilités de **savoir** d'un objet peuvent être
 - **Connaître** les données privées encapsulées
 - Un objet profil qui contient un code secret qui peut être validé mais pas lu
 - **Connaître** les objets connexes
 - Relation entre la classe qui reçoit un commandement et la classe qui l'interprète
 - **Connaître** les éléments qu'il peut dériver ou calculer
 - Comment une règle peut-elle « transférer » la règle suivante (ou les informations pour la déterminer)?
- **CPR : Permet de définir une communauté d'objets qui collaborent! C'est une philosophie**

8

Une exception en C++

```
int W000Adapter::attaquer(int x, int y, int xEnnemi, int yEnnemi)
{
    x = 0;
    x = 2 / x;
    if (!estFonctionnelle())
        return;
    if (w000->g
        return;
}

void W000Adapte
{
    ...
}
```

Exception non gérée

Exception non gérée à 0x00007FF62ADF456D dans XBot.exe : 0xC0000094: Integer division by zero.

[Afficher la pile des appels](#) | [Copier les détails](#) | [Démarrer la session Live Share](#)

Paramètres d'exception

Pile des appels	
Nom	Lang
XBot.exe!W000Adapter::attaquer(int x, int y, int xEnnemi, int yEnnemi) Ligne 41	C++
XBot.exe!Arene::tourCombat(Arene::Combattant ** attaquants, Arene::Combattant ** defenseurs) Ligne 116	C++
XBot.exe!Arene::debuterCombat() Ligne 95	C++
XBot.exe!main() Ligne 432	C++
[Code externe]	

9

Une exception en .NET(C#)

```
Unhandled exception. System.AggregateException: One or more errors occurred. (Value cannot be null.
(Parameter 'password'))
---> System.ArgumentNullException: Value cannot be null. (Parameter 'password')
   at Microsoft.AspNetCore.Identity.UserManager`1.CreateAsync(TUser user, String password)
   at Capsule.Data.SeedData.EnsureUser(IServiceProvider serviceProvider, String testUserPw, String U
serName) in C:\Users\ericdallaire\Desktop\exemple\CapsuleAuth-main\Capsule\Data\SeedData.cs:line 36
   at Capsule.Data.SeedData.Initialize(IServiceProvider serviceProvider, String testUserPw) in C:\Us
ers\ericdallaire\Desktop\exemple\CapsuleAuth-main\Capsule\Data\SeedData.cs:line 16
--- End of inner exception stack trace ---
   at System.Threading.Tasks.Task.ThrowIfExceptional(Boolean includeTaskCanceledExceptions)
   at System.Threading.Tasks.Task.Wait(Int32 millisecondsTimeout, CancellationToken cancellationToke
n)
   at System.Threading.Tasks.Task.Wait()
   at Capsule.Program.Main(String[] args) in C:\Users\ericdallaire\Desktop\exemple\CapsuleAuth-main\
Capsule\Program.cs:line 54


Sortie de C:\Users\ericdallaire\Desktop\exemple\CapsuleAuth-main\Capsule\bin\Debug\net6.0\Capsule.ex
e (processus 8972). Code : -532462766.
```

10


Facteurs de considération



Classes vs sous-systèmes vs systèmes



Encapsulation et modularité



Niveau d'abstraction et de personnalisation

11

Facteurs de considération



Packages



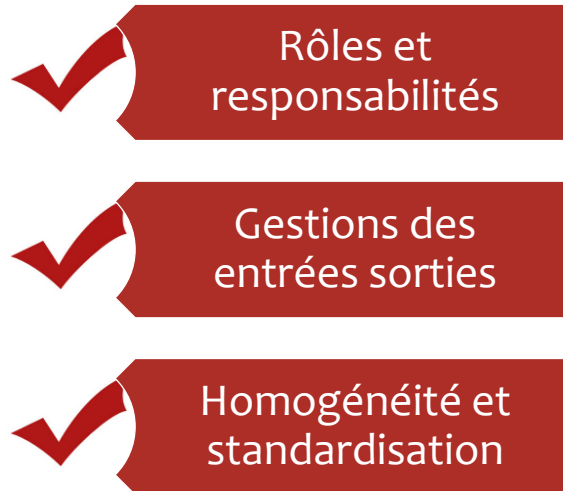
Communication et relation



Message - objet de communication

12

Facteurs de considération



13

Démarrage des applications

- Un système a toujours une méthode qui permet le **démarrage** de l'application
 - Quels objets sont initialisés au démarrage
 - Comment le système va-t-il réagir au monde extérieur!
- Cette facette peut être développée en dernier lieu
 - Pourquoi?
- Réflexion
 - Comment on pourra démarrer et rendre persistant un système qui n'a pas d'interface graphique?
 - A-t-on besoin de persistance? Gère-t-on simplement des flux?

14

Patrons de conception

- Définition
 - Principes généraux et solutions qui guident la conception logicielle
- Il existent de nombreux patrons qui possèdent
 - Un nom
 - Un problème courant qui est clairement identifié
 - Une solution simple et standard (implémenté dans divers langages)
 - Des contextes d'utilisation
 - Des avantages et des inconvénients
- En fait
 - Un patron de conception n'apporte pas de nouvelles solutions
 - Un expert les trouvera simplistes mais bien utiles ;-)
 - Les plus connus : **GoF** et **GRASP**

15

GRASP

- GRASP
 - « **General Responsibility Assignment Software Patterns** »
 - Aide pédagogique destinée à vous aider à **comprendre** l'essentiel de la **conception orientée objet** et d'**appliquer** ses raisonnements de façon méthodique, rationnelle et explicable.
 - GRASP vous permettra d'améliorer et de mieux définir vos diagrammes UML
- « the critical design tool for software development is a **mind well educated in design principles**. It is not the UML or any other technology »

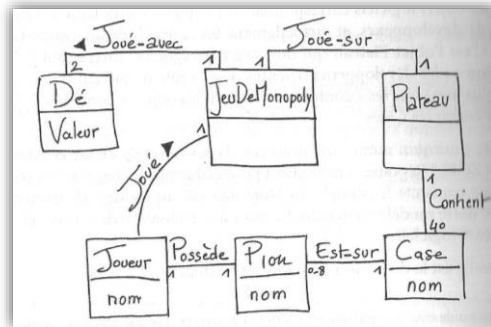
Craig Larman

16

GRASP

• Les patrons GRASP

- Responsabilités
- Créateur
- Expert
- Faible couplage
- Contrôleur
- Forte cohésion
- Polymorphisme
- Indirection
- Fabrication pure



17

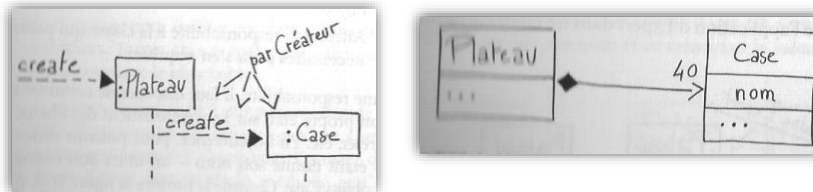
GRASP - Créateur

Nom	Créateur
Problème	Qui crée un A?
Solution	<p>Affecter à la classe B la responsabilité de créer une instance de la classe A si une ou plusieurs des conditions suivantes est vraie</p> <ul style="list-style-type: none"> • Contient ou agrège des objets A • Enregistre des objets A • Utilise étroitement des objets A • Possède des données d'initialisation des objets A

18

GRASP - Créateur

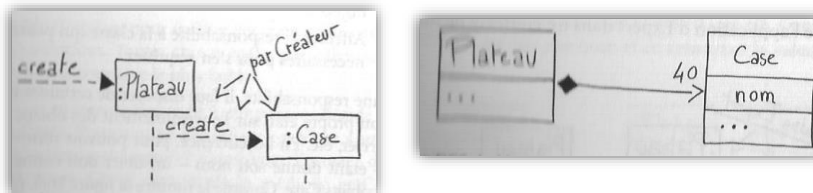
- Problème : qui crée l'objet « Case »
 - Tous les objets pourraient le créer...
 - Pourquoi on choisi un objet en particulier
 - On possède un modèle mental du domaine, une représentation
- Objectif
 - Réduire le décalage des représentations (le bon sens, la logique)



19

GRASP - Créateur

- Réflexion
 - La plateau « contient » les cases (**composition**) mais ne doit pas obligatoirement être la classe qui les « crée ». Elle pourrait seulement les **recevoir** d'une manufacture???
 - Que doit-on **connaître** pour créer les classes? Est-ce que l'ordre des cases est prédéfini? Sont-elles dynamiques? Paramétrisables?



20

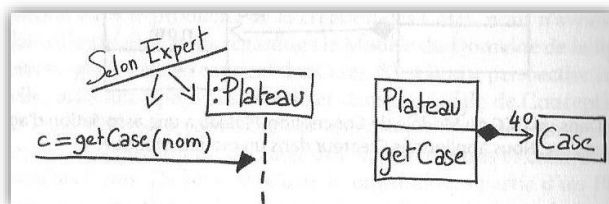
GRASP - Expert

Nom	Expert en information
Problème	Quel est le principe général d'affectation des responsabilités aux objets?
Solution	Affecter la responsabilité à la classe qui possède les informations nécessaires pour s'en acquitter. Il lui faut connaître <ul style="list-style-type: none"> • Les autres objets • Les états et contextes d'utilisation • Des informations sur l'environnement • Ce qu'il aura à faire et pourrait faire

21

GRASP - Expert

- Problème : Qui **connaît** l'objet « Case »
 - Qui est responsable de la case
 - Si j'ai besoin de la case en particulier, à qui je dois parler et de quelle manière?
 - On doit aussi réfléchir aux permissions sur l'objet (encapsulation)
- Objectif
 - Définir les **responsabilités**



22

GRASP – Faible couplage

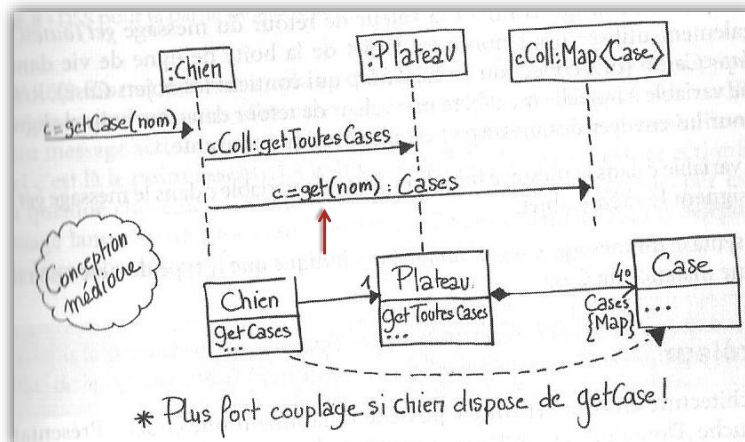
- Nom **Faible couplage**
- Problème Comment réduire l'impact des modifications
- Solution Affecter la responsabilité de façon à éviter tout couplage inutile.
- Appliquer ce principe pour évaluer les options disponibles
 - Un couplage est une dépendance vers un objet. En gros, si A doit être modifié lorsque B est modifié, A est couplé à B.

(ABSTRACTION – INTERFACE – DELEGATE – POLYMORPHISME – DLL – sous système)

23

GRASP – Faible couplage

- Problème : Pourquoi un plateau plutôt qu'un chien?



24

GRASP - Contrôleur

Nom	Contrôleur
Problème	Quel est le premier objet au-delà de la couche présentation qui reçoit et coordonne une opération système
Solution	Affecter la responsabilité à un objet représentant l'un de ces choix <ul style="list-style-type: none"> • Objet représentant le système, un objet racine • Objet répondant défini spécifiquement lors de l'analyse (ex : gestionnaire de services d'Android)

25

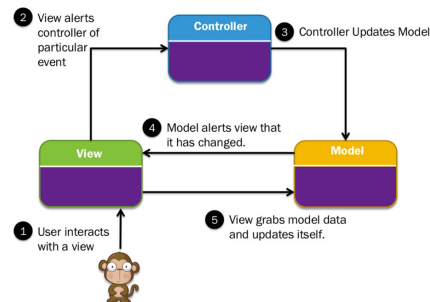
GRASP - Contrôleur

- Problème : Qui est le contrôleur de l'opération « lancer partie »

- Une classe qui représente le jeu lui-même
- L'utilisateur « joue au **jeu de Monopoly** »

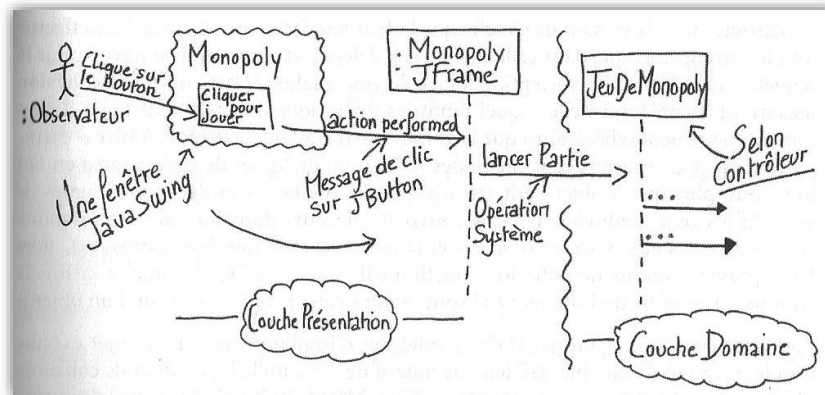
- Objectif

- Définir comment connecter la couche présentation à la couche application



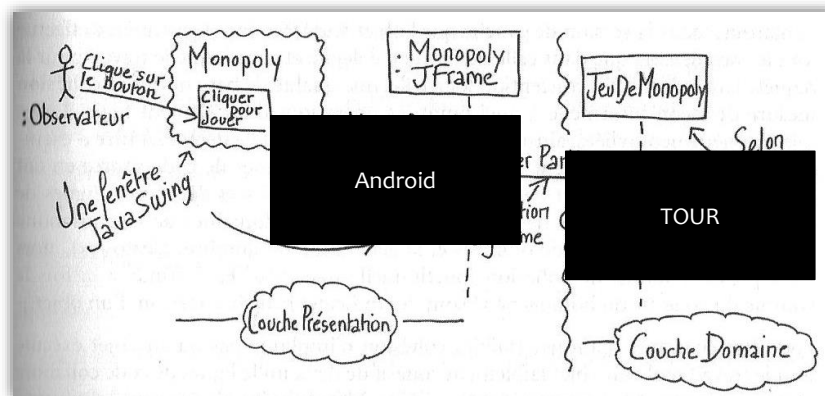
26

GRASP - Contrôleur



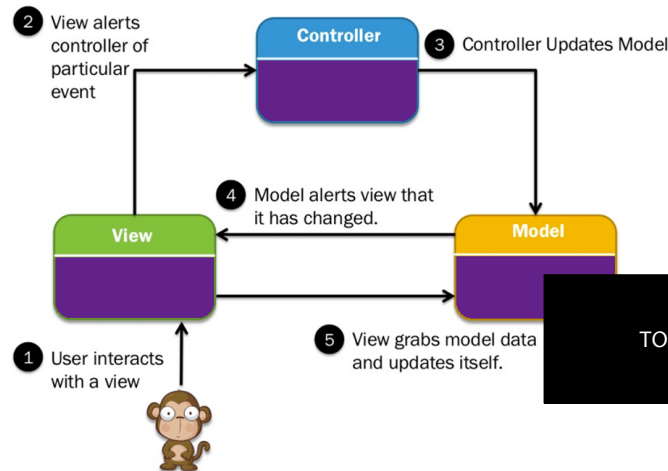
27

GRASP - Contrôleur



28

GRASP - Contrôleur



29

GRASP – Forte cohésion

Nom	Forte cohésion
Problème	Comment s'assurer que les objets restent compréhensibles et faciles à gérer tout en contribuant au faible couplage
Solution	<p>Affecter les responsabilités pour que la cohésion demeure élevée</p> <ul style="list-style-type: none"> • Appliquer ce principe pour évaluer les diverses solutions • Permet de créer une série de couches d'abstraction entre les classes, les sous-systèmes et système

30

GRASP – Forte cohésion

