

2. Description (et classification) des design patterns

Classification des patterns (GoF)

- 23 *patterns* du GoF
- Différentes catégories existent (classification)
 - Classer l'ensemble des patterns pour faciliter
 - l'accès
 - la comparaison
 - l'utilisation
 - Proposition de 2 axes de classification orthogonaux



Classification des patterns (GoF)

- Axe 1 : classification selon la finalité
 - ☛ Modèles « créateurs »
 - Visent l'instanciation des objets et permettent le découplage entre l'objet qui a besoin d'instancier et les instances
 - ☛ Modèles « structurels »
 - Visent la composition de classes ou d'objets pour former des structures
 - *NB : on peut aussi composer des patterns mais c'est un autre pb...*
 - ☛ Modèles « comportementaux »
 - Visent l'organisation de l'interaction entre des classes ou des objets ainsi que la distribution des responsabilités
 - *p. ex. Patron de Méthode, Stratégie*
- Distinguo entre catégories non trivial !
 - ☛ • « L'intention » du pattern est un critère déterminant (cf. diapo 2/5 du « canevas » de description)



Classification des patterns (GoF)

- Axe 2 : classification selon le « niveau » d'application
 - ☛ modèles de niveau « classe »
 - Définissent les relations entre classes au moyen de l'héritage et de l'association (statiquement, à la compilation)
 - *Patron de méthode*
 - ☛ modèles de niveau « objet »
 - Définissent les relations entre objets au moyen de la délégation (dynamiquement, à l'exécution)
 - La plus grande partie des modèles entrent dans cette catégorie
 - *Stratégie*

Classification des 23 patterns du GoF

		Rôle		
		Créateur	Structurel	Comportemental
Niveau / Domaine	Classe	Fabrication	Adaptateur (de classe)	Interprète Patron de méthode
	Objet	Fabrique abstraite Monteur Prototype Singleton	Adaptateur (d'objet) Pont Composite Décorateur Façade Poids mouche Procuration (Proxy)	Chaîne de responsabilités Commande Itérateur Médiateur Memento Observateur Etat Stratégie Visiteur

Classification des patterns (Metsker & Wake)

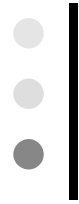
- Une autre classification des patterns du GoF...
- Patterns d'interface
 - Pour définir ou redéfinir l'accès aux méthodes d'une classe ou d'un groupe de classes
 - Et répondre à un besoin auquel les seules interfaces (Java) ne permettent pas de répondre
 - *Adaptateur, Façade, Composite, Pont*
- Patterns de responsabilité
 - Pour définir (centraliser, transmettre, limiter) la responsabilité de certains objets
 - *Singleton, Observateur, Médiateur, Proxy, Chaîne de responsabilités, Poids mouche*

Classification des patterns (Metsker & Wake)

- Patterns de construction
 - Pour permettre la création d'objet dans le cas où l'utilisation de constructeurs ordinaires ne suffit ou ne convient pas
 - *Monteur, Fabrication, Fabrique abstraite, Prototype, Memento*
- Patterns d'opérations
 - Pour permettre le contrôle de l'invocation de méthodes
 - *Patron de méthode, Etat, Stratégie, Commande, Interprète*
- Patterns d'extensions
 - Pour permettre l'extension de code existant (ajout de fonctionnalités)
 - *Décorateur, Itérateur, Visiteur*

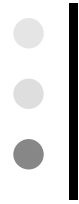
Description : principaux éléments d'un modèle

- Nom
 - Identifiant d'un concept
- Problème et contexte
 - « *forces* »
 - but à atteindre
 - ensemble des contraintes
 - Situations dans lesquelles le modèle s'applique (contexte)
- Solution
 - Éléments du modèle de conception
- Conséquences et réalisation
 - Effets de l'application du modèle sur la conception
 - Mise en œuvre



Canevas de description en 13 pts (GoF) 1/5

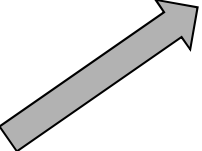
- Nom (identification)
 1. Nom du modèle et classification
 - Vocabulaire
 - Catégorie du modèle
 2. Alias
 - Autre appellation (synonyme)



Canevas de description en 13 pts (GoF) 2/5

- Problème et contexte

- 3. Intention ~ quoi

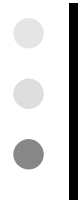
- 
- But et raison d'être (« forces »)
 - Description du problème de conception visé
 - Description générale de ce que fait effectivement le modèle (grandes lignes)

- 4. Motivation (justification) ~ pourquoi

- Illustration de l'intérêt du modèle et de son apport (comment le modèle résout le problème) au moyen d'un scénario concret

- 5. Indications d'utilisation ~ quand

- Cas et situations qui justifient l'utilisation du modèle et dans lesquels il est avantageux de l'utiliser



Canevas de description en 13 pts (GoF) 3/5

- Solution

- 6. Structure

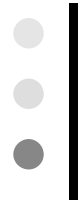
- Diagramme de classes qui représente graphiquement les relations entre les participants (en UML)

- 7. Constituants (ou participants)

- Classes et interfaces, ainsi que leurs rôles et responsabilités respectives

- 8. Collaborations

- Comment les participants collaborent (diagrammes de...)



Canevas de description en 13 pts (GoF) 4/5

- Conséquences et réalisation

- 9. Conséquences

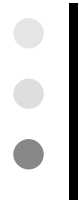
- Impact de l'utilisation du modèle sur l'architecture et les propriétés non fonctionnelles (effets positifs ou négatifs)
 - Marge de manœuvre laissée au concepteur et compromis nécessaires

- 10. Implémentation

- Techniques d'implémentation à employer
 - Spécificités concernant certains langages de programmation

- 11. Exemples de code

- Extraits de programmes (en Java, C++, etc.)



Canevas de description en 13 pts (GoF) 5/5

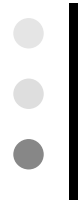
- Compléments

- 12. Utilisations remarquables

- Exemples significatifs de l'utilisation du modèle dans des systèmes existants

- 13. Modèles apparentés (liens avec les autres modèles)

- Modèles voisins et différences importantes avec ceux-ci
 - Autres modèles avec lesquels il peut être utilisé conjointement



Comment choisir et utiliser un design pattern ?

1. À partir d'un catalogue de patterns (description générale)
 - Identifier ceux qui sont susceptibles de répondre au problème
2. À partir de la description détaillée de chaque pattern identifié
 - Etudier en détail l'adéquation du design pattern (son intention) au problème afin de déterminer s'il le résout (en l'adaptant plus ou moins)
3. Choisir puis appliquer le pattern
 - Renommer les participants et les méthodes (rendre les noms explicites dans le cadre de l'application)
 - Adapter la structure générique pour répondre aux besoins et aux contraintes spécifiques de l'application
4. Documenter la solution !