

Design Patterns pour les Systèmes Interactifs



Philippe Palanque
ICS-IRIT

palanque@irit.fr

<http://www.irit.fr/ICS/palanque>





Introduction

- Un concept pour augmenter la réutilisabilité du logiciel orienté-objet
- Donner des archétypes de solution à des problèmes récurrents dans la conception de logiciel
- Solutions issues de l'expérience dans des projets réels, validées par l'usage
 - « On n'invente pas un Pattern, on le découvre »



Inspiration Architecturale

- « Un pattern décrit un problème qui se présente fréquemment dans notre environnement, et décrit l'essence d'une solution à ce problème, de telle sorte que l'on puisse réutiliser cette solution un million de fois sans jamais la refaire deux fois de la même manière »
 - Christopher Alexander



Bibliographie

- Design Patterns, Elements of Reusable Object-Oriented Software
 - E. Gamma, R. Helm, R. Johnson, J. Vlissides, Addison-Wesley, le «Gang of Four», 1995
- A System of Patterns : Pattern-oriented software architecture
 - Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad, and M. A Stal, Wiley 1996
- Smalltalk-80: the language and its implementations
 - Goldberg, A. and Robson, D.. Addison Wesley; 1983.
- Java



Éléments d'un Pattern

- Nom du pattern
 - augmenter le vocabulaire du design, augmenter le niveau d'abstraction
- Description du problème
 - contexte d'application
- Solution
 - en termes d'un modèle générique (la plupart du temps OO)
- Conséquences
 - avantages, inconvénients, compromis



Les patterns ne sont pas

- Des bibliothèques de classes réutilisables
 - listes chaînées, arbres, ...
- Des applications toutes faites
 - traitent uniquement un sous-problème bien défini
- Des classes génériques (templates)



Design Pattern 1: MVC

Goldberg, A. and Robson, D..
Addison Wesley; 1983 Smalltalk

Le Design Pattern MVC par l'exemple

The screenshot shows a Microsoft PowerPoint presentation titled "Design Pattern MVC par l'exemple". The presentation is displayed in a window titled "DesignPatternMVC (Mode de compatibilité) - Microsoft PowerPoint". The slide content includes the title "Design Pattern MVC par l'exemple" and a large empty space for a diagram. On the left side, there is a sidebar with several thumbnails of slides, including "Le Design Pattern MVC par l'exemple", "Le Pattern Model", "Le Pattern Model + View", "Objectif", "Structure générale", "Volume", and "Séquence Diagram". A small window titled "Contrôle de volume" is open over the sidebar, showing volume controls for four speakers. The bottom of the slide has a text box that says "Cliquez pour ajouter des commentaires". The right side of the screen shows the "Images cliquables" pane with search options.

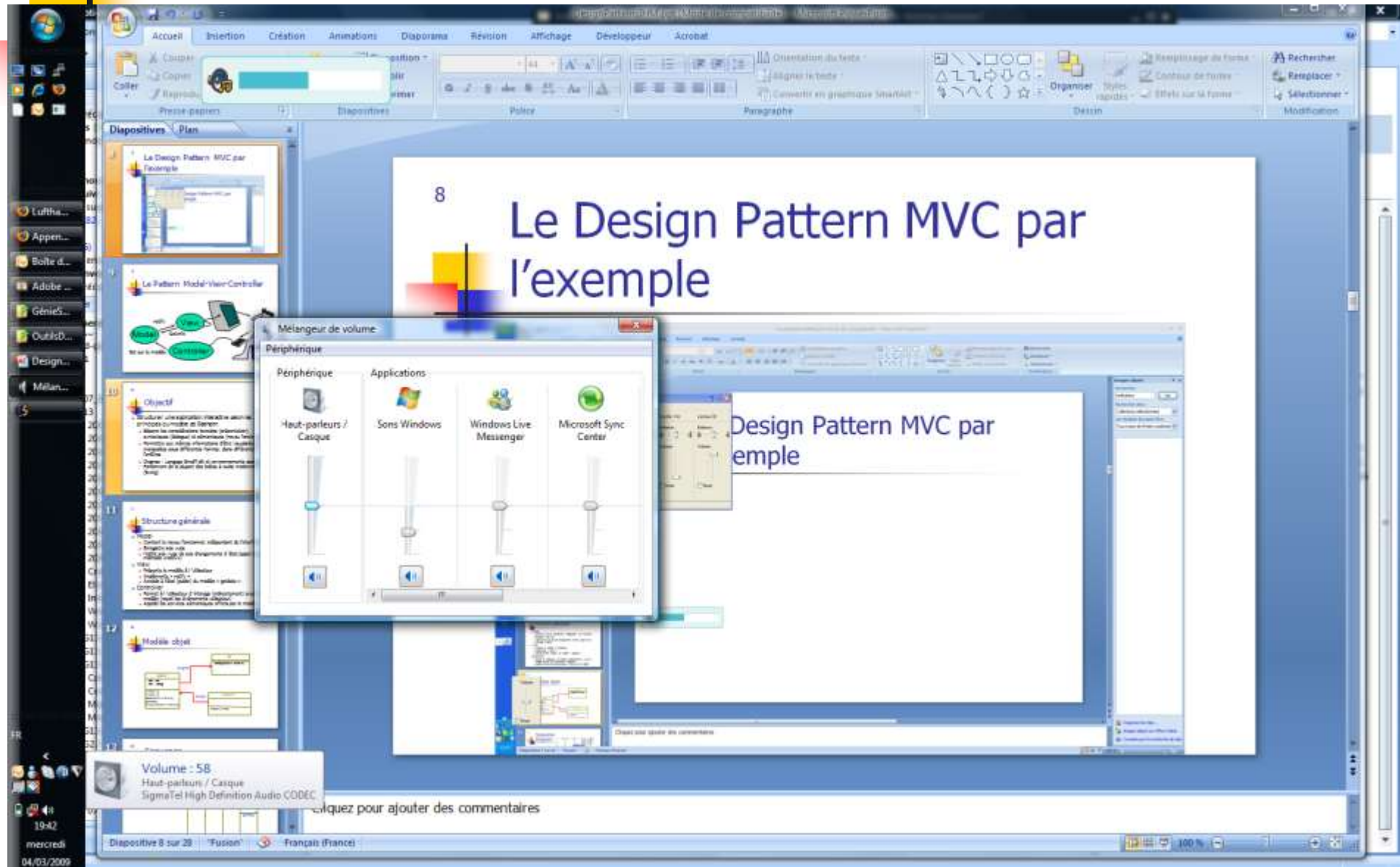
9 Le Design Pattern MVC par l'exemple

The image shows a screenshot of a presentation slide titled "Le Design Pattern MVC par l'exemple" (The MVC Design Pattern by example). The slide is displayed in a presentation software window, likely Microsoft PowerPoint, with a ribbon menu at the top showing tabs like "Accueil", "Insertion", "Création", "Animations", "Diaporama", "Révision", "Affichage", "Développeur", and "Acrobat". The slide content includes a title, a small graphic of overlapping colored squares, and a list of bullet points under the heading "Objectif".

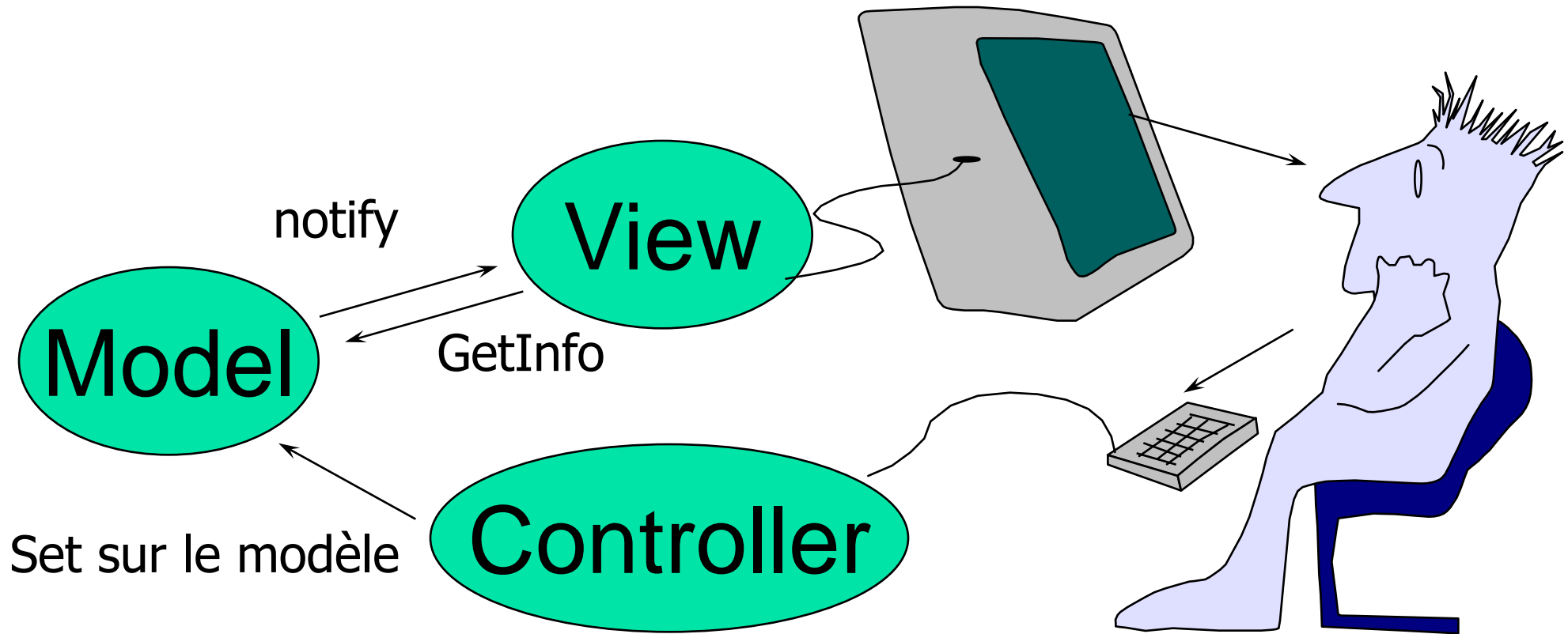
Overlaid on the presentation is a "Mélangeur de volume" (Volume Mixer) window. It displays volume sliders for various applications and the system speaker. The applications listed are "Haut-parleurs / Casque", "Sons Windows", "Windows Live Messenger", and "Microsoft Sync Center". Each application has a volume slider and a speaker icon.

The presentation software interface also shows a sidebar with a list of slides, including "Le Design Pattern MVC par l'exemple", "C'est super mais faut pas rêver", "Le Pattern Model-View-Controller", "Objectif", and "Structure générale". The status bar at the bottom indicates "Diapositive 9 sur 30", "Fusion", and "Français (France)".

C'est super mais faut pas rêver



Le Pattern Model-View-Controller

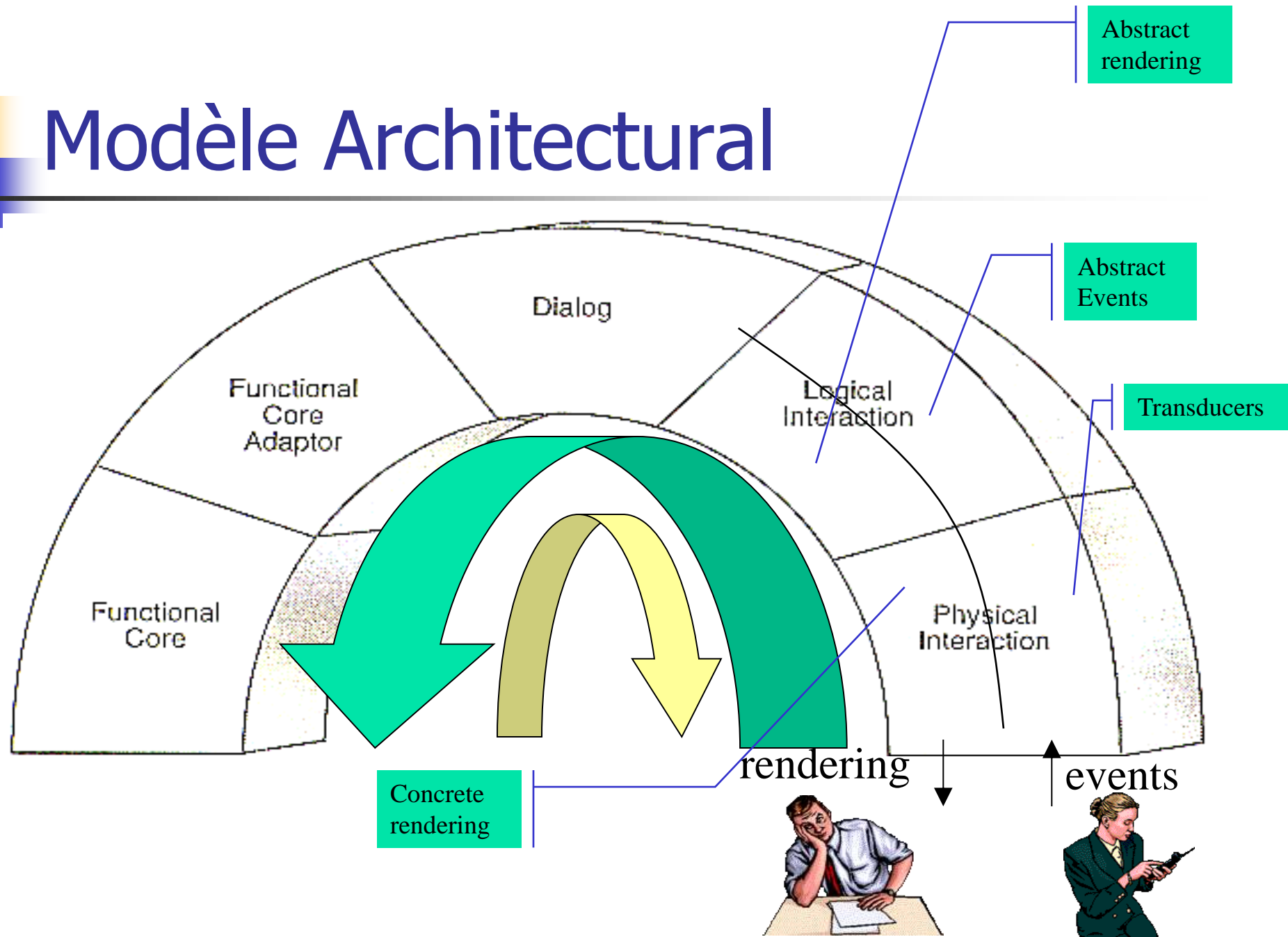




Objectif

- Structurer une application interactive selon les principes du modèle ARCH
 - Séparer les considérations lexicales (présentation), syntaxiques (dialogue) et sémantiques (noyau fonctionnel)
 - Permettre aux mêmes informations d'être visualisées et manipulées sous différentes formes, dans différentes fenêtres
 - Différents chemins entre input et output
 - Origines : Langage SmallTalk et environnements associés. Fondement de la plupart des boîtes à outils modernes (Swing)

Modèle Architectural





Structure générale

■ Model

- Contient le noyau fonctionnel, indépendant de l'interface
- Enregistre ses vues
- Notifie ses vues de ses changements d'état (appel de la méthode «notify»)

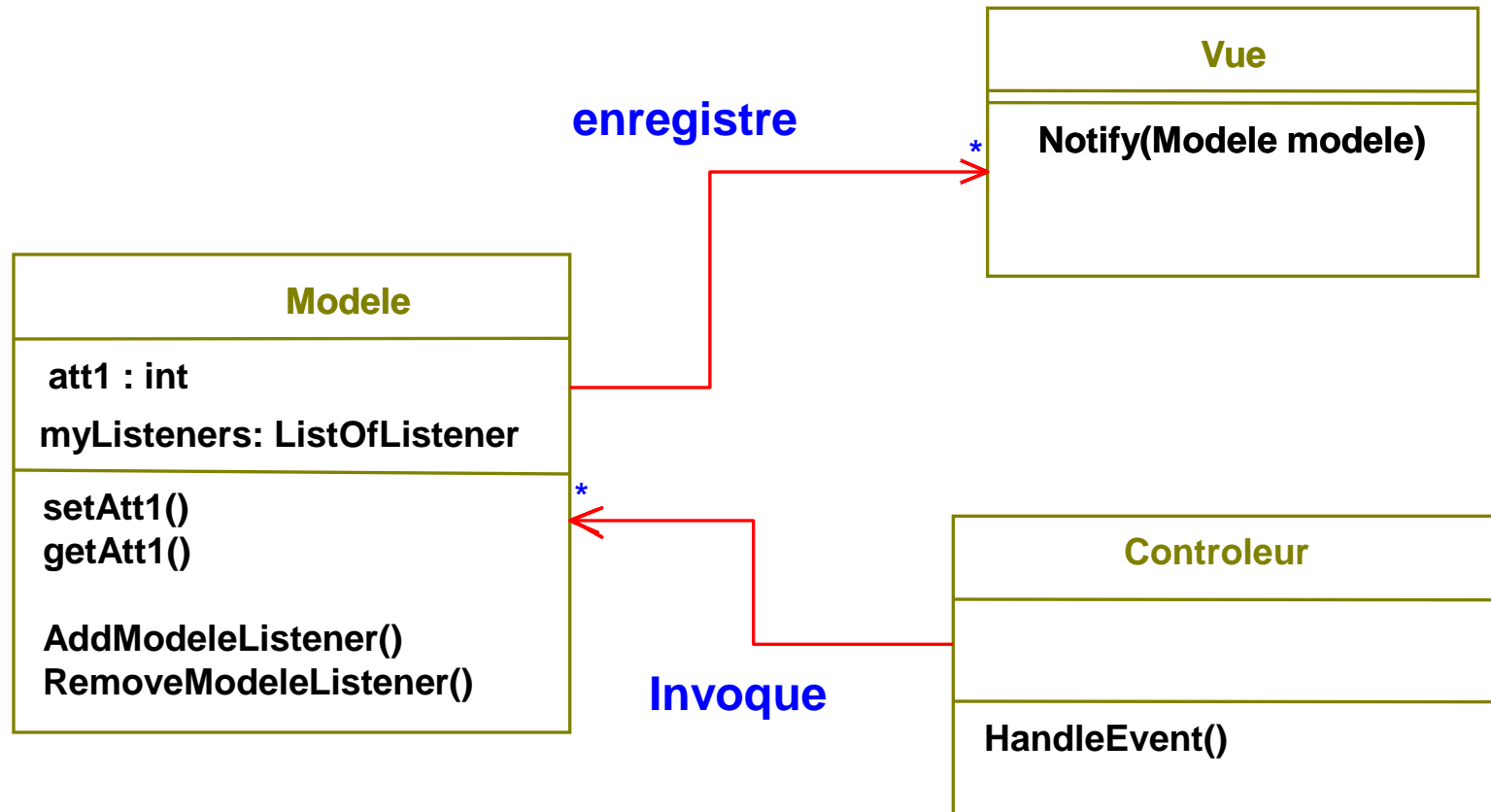
■ View

- Présente le modèle à l'utilisateur
- Implémente «notify»
- Accède à l'état (public) du modèle « getdata »

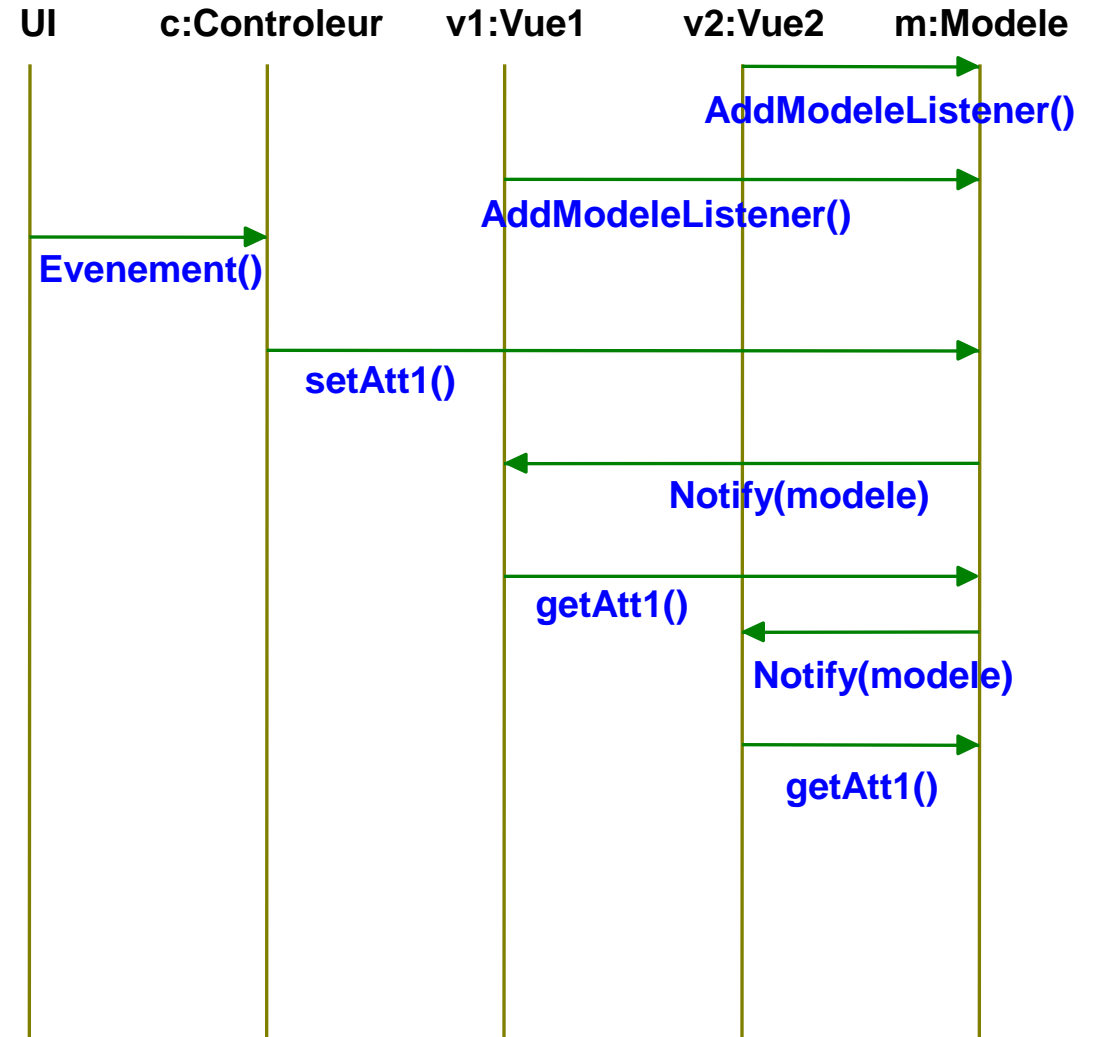
■ Controller

- Permet à l'utilisateur d'interagir (**indirectement**) avec le modèle (reçoit les événements utilisateur)
- Appelle les services sémantiques offerts par le modèle (modifs)

Modèle objet



Sequence Diagram



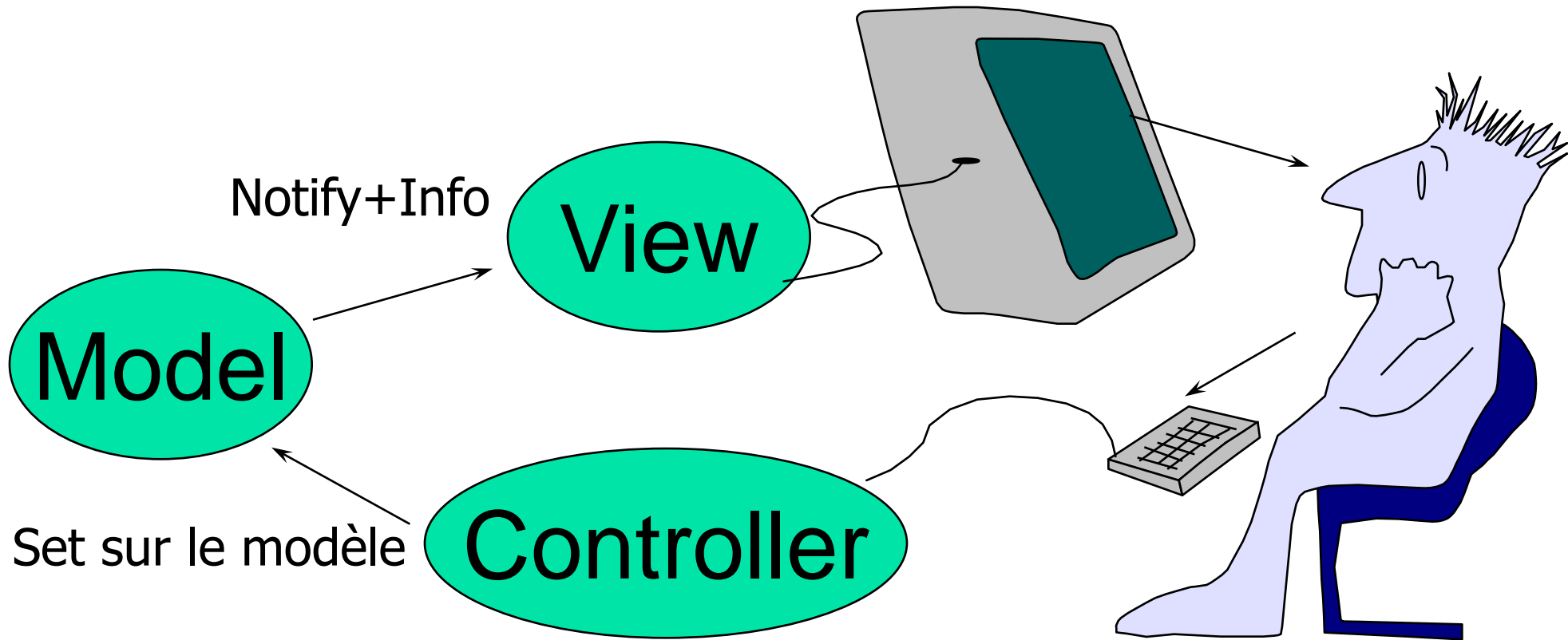


Conséquences

- Avantages
 - Sépare les données de leur présentation et manipulations
 - Facile de présenter différentes vues du même objet et de les garder synchronisées
 - Facile d'ajouter une nouvelle représentation (ou d'en enlever)
- Inconvénients
 - Multiplication des appels à la méthode «notify»
 - D'où inefficacité de l'accès au modèle
 - En général couplage fort entre view et controller
 - Impossible en général de réutiliser un contrôleur indépendamment de sa vue e.g. textbox, ...
 - Modification modèle entraine modification du controller (les appels)
- Règle très importante: ce ne sont que des rôles
- Variantes
 - ALV (Abstraction Link View), PAC (Presentation Abstraction Control Coutaz 1987), ...

Le Pattern Model-View-Controller

«relaxed»





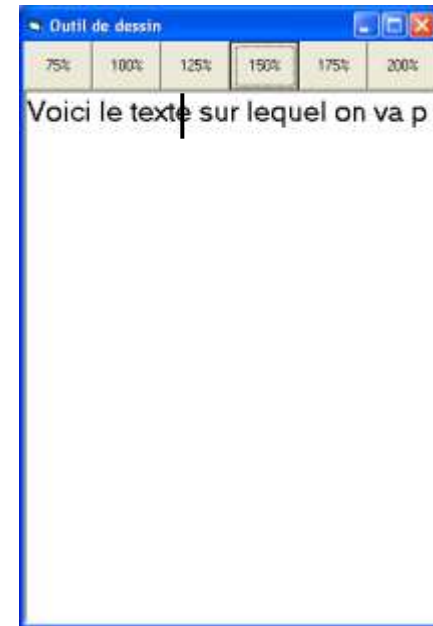
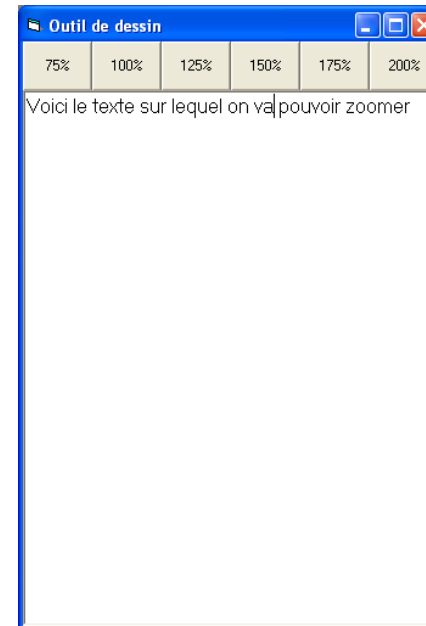
Exemple particulier

- Où sont les éléments M, V et C dans un la fenêtre?
- Décrivez l'architecture ainsi que les séquences d'appel de méthodes

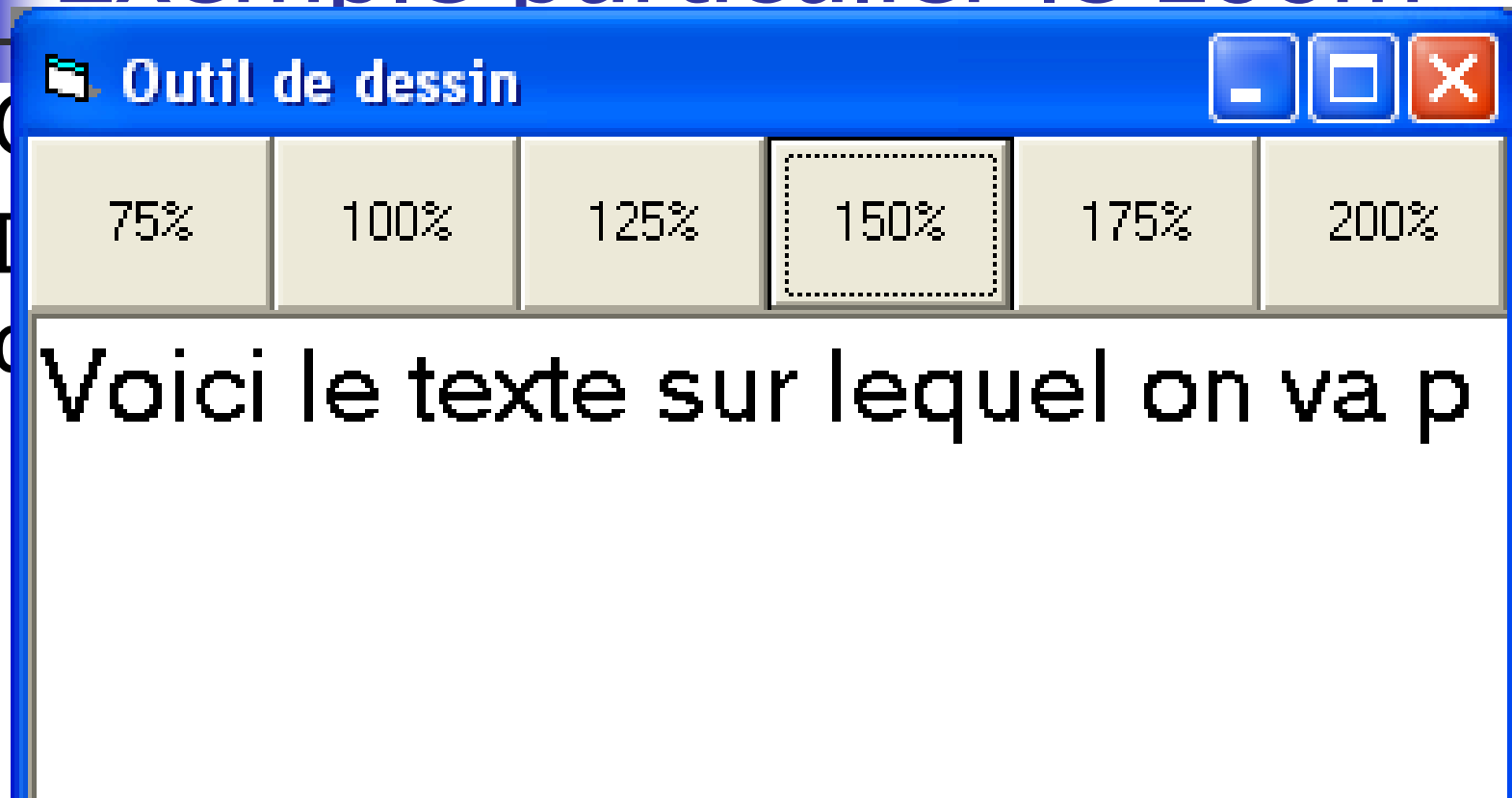


Exemple particulier le zoom

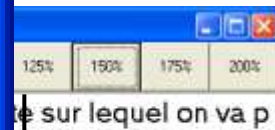
- Comment gérer plusieurs contrôleurs
- Décrivez l'architecture ainsi que les séquences d'appel de méthodes



Exemple particulier le zoom



nces



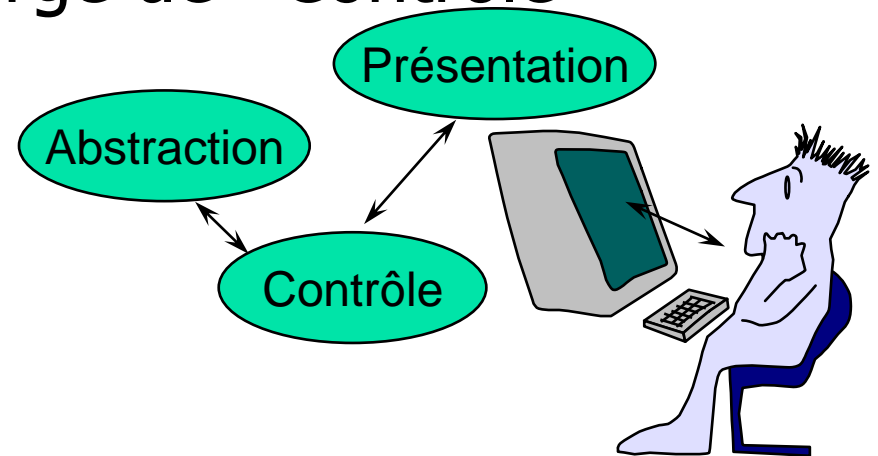


Exemple particulier le zoom



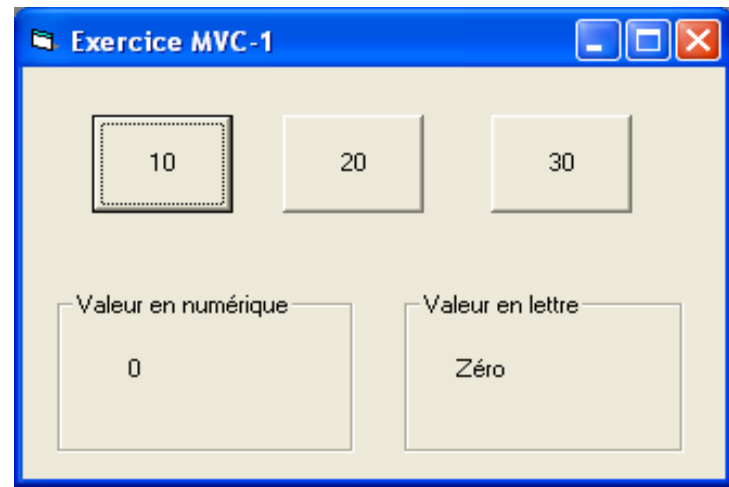
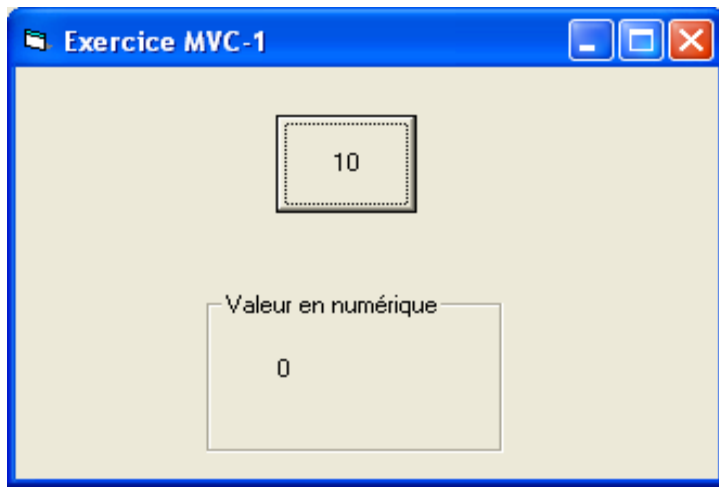
Présentation-Abstraction-Contrôle (Coutaz 1987)

- Mêmes objectifs, factorisation différente
 - «Abstraction» joue le rôle de «Modèle»
 - On regroupe «Vue» et «Contrôleur» en une seule classe «Présentation»
 - L'enregistrement des présentations et leur notification est à la charge de «Contrôle»



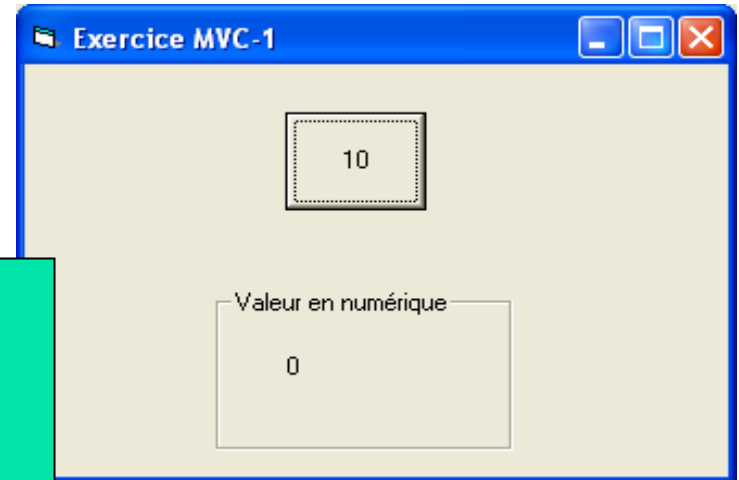
Exercice MVC

- 2 applications avec des modifications minimales grâce à une bonne structuration du code
- Objectif accroître la modifiabilité de l'application



Exercice MVC

- Cas 1





Exercice MVC : Interfaces

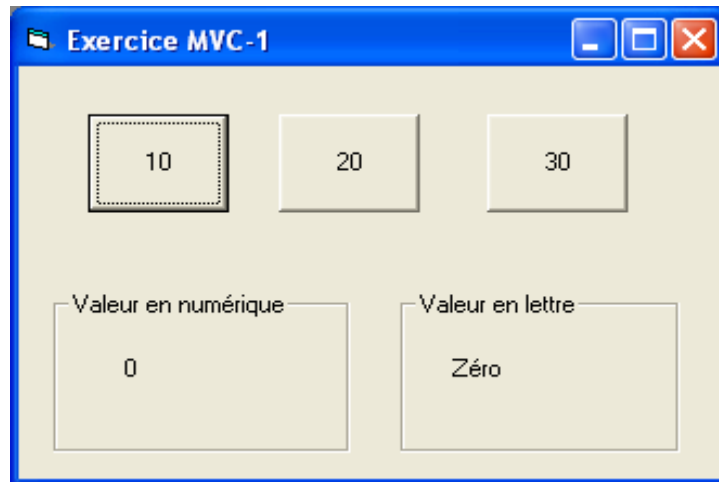
- 2 interfaces
 - IVue : offre une méthode notification()
 - IModel : offre une méthode addVue(IVue vue)
- Les vues sont "implements" de l'interface IVue
- Les modèles sont "implements" de l'interface IModel
- Objectif
 - Rapidité de création de nouvelles vues
 - Cohérence « forcée » des vues
 - Ne pas faire plusieurs fois la même chose



Exercice MVC : Classes

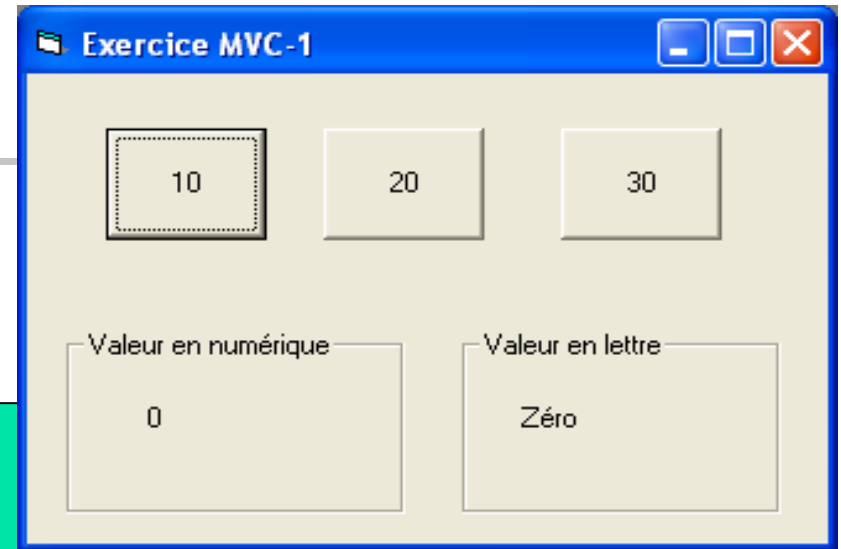
- VueLabel1
 - A une référence vers le modèle
 - A une référence vers la fenêtre où elle s'affiche (pas nécessaire)
- ModelString
 - Contient un attribut (String valeur)
 - Offre l'accessor setValeur (String chaine) : appelé par les contrôleurs
 - Offre l'accessor getValeur (returns valeur) : appelé par les vues
 - Connait chacune des vues (géré dans une LinkedList)
 - Gère les vues en offrant un abonnement/désabonnement
 - Possède un itérateur pour la notification de toutes les vues
- JButton1: contrôleur direct (pas de classe supplémentaire)
 - Connait l'instance de ModelString
 - Appelle setValeur (dans l'événement handler ou "actionPerformed")

Exercice MVC : 3 contrôleurs et 2 vues du même modèle



Exercice MVC

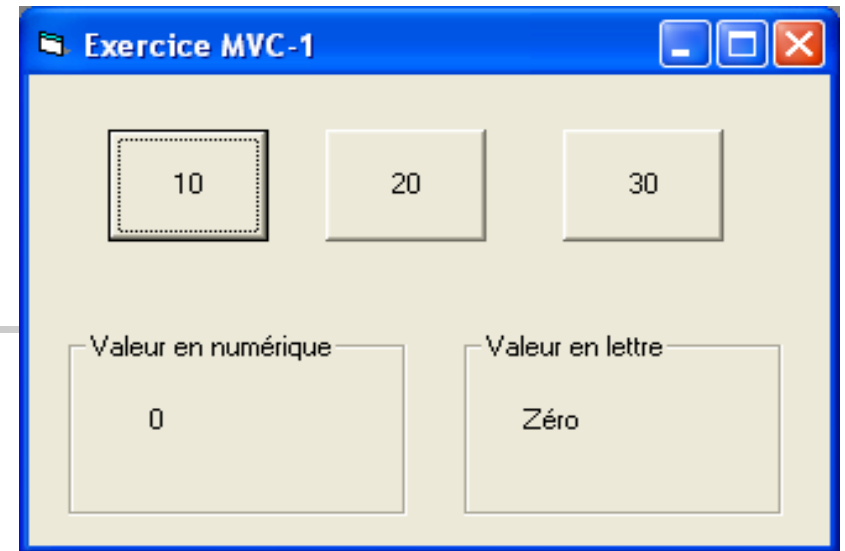
■ Cas 2



The screenshot shows a Windows application window titled "Exercice MVC-1". The window contains a simple MVC interface with three buttons at the top labeled "10", "20", and "30". The "10" button is highlighted with a dashed border. Below these buttons are two text boxes. The left text box is labeled "Valeur en numérique" and contains the value "0". The right text box is labeled "Valeur en lettre" and contains the value "Zéro".

Exercice MVC

■ Cas 2



The screenshot shows a Windows application window titled "Exercice MVC-1". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area is light gray and contains three buttons at the top labeled "10", "20", and "30". Below these buttons are two text boxes. The left text box is labeled "Valeur en numérique" and contains the text "0". The right text box is labeled "Valeur en lettre" and contains the text "Zéro".

Exercice MVC-3

- Mélange automates et design patterns

The screenshot shows a Java Swing window titled "Exercice MVC-1". The window contains three buttons at the top with the values "10", "20", and "30". The button with "20" is currently selected, indicated by a dashed border. Below these buttons are two text input fields. The left field is labeled "Valeur en numérique" and contains the text "10". The right field is labeled "Valeur en texte" and contains the text "Dix".