

Modèle MVC

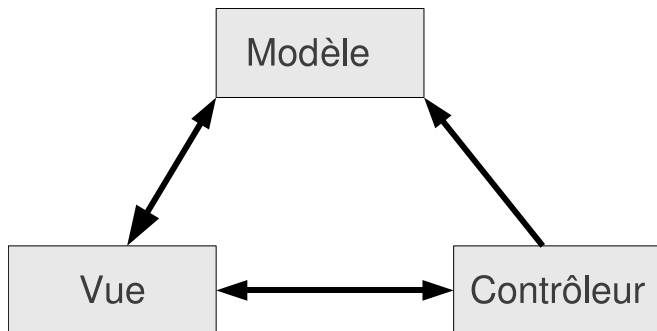
Michel C. Desmarais

Génie informatique et logiciel
École Polytechnique de Montréal

29 octobre 2017

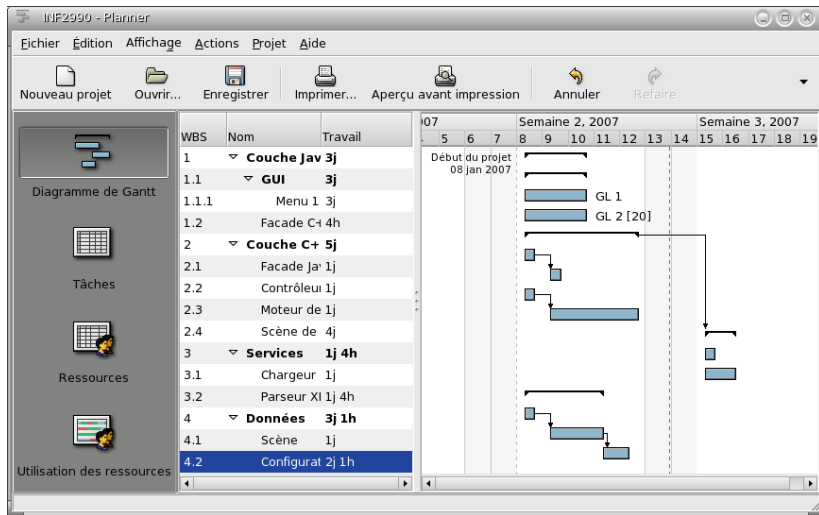
Modèle MVC

Architecture modèle-vue-contrôleur



Exemple en gestion de projet

Vue Gantt



Exemple en gestion de projet

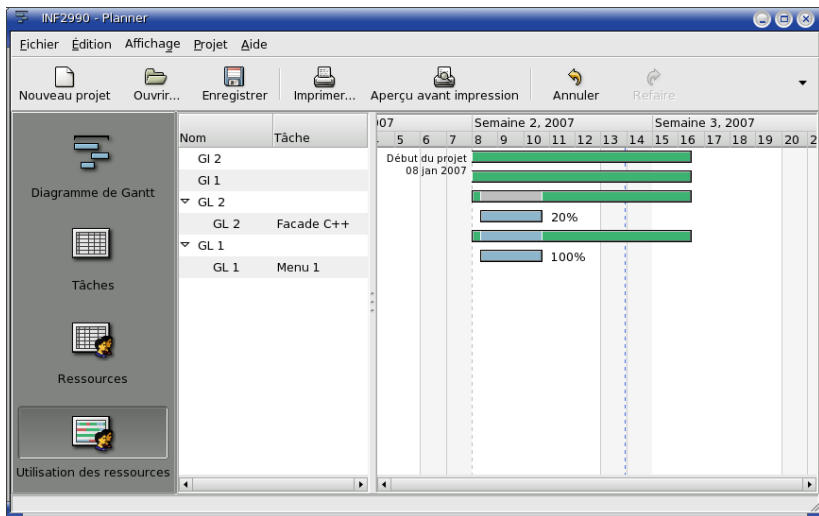
Vue Tâches

The screenshot shows the 'INF2990 - Planner' application window. The menu bar includes 'Fichier', 'Édition', 'Affichage', 'Actions', 'Projet', and 'Aide'. The toolbar contains icons for 'Nouveau projet', 'Ouvrir...', 'Enregistrer', 'Imprimer...', 'Aperçu avant impression', 'Annuler', and 'Refaire'. The left sidebar has four main sections: 'Diagramme de Gantt', 'Tâches' (selected), 'Ressources', and 'Utilisation des ressources'. The main area displays a task list table.

WBS	Nom	Démarrer	Terminer	Travail	Durée	Mou	Coût	Assigned to
1	▼ Couche Jav	jan 8	jan 10	3j	3j	3j 4h	28,8	
1.1	▼ GUI	jan 8	jan 10	3j	3j	3j 4h	24	
1.1.1	Menu 1	jan 8	jan 10	3j	3j	3j 4h	24	GL 1
1.2	Facade C+	jan 8	jan 10	4h	3j	5h	4,8	GL 2
2	▼ Couche C+	jan 8	jan 12	5j	5j		0	
2.1	Facade Ja	jan 8	jan 8	1j	1j	3j	0	
2.2	Contrôleu	jan 9	jan 9	1j	1j	3j	0	
2.3	Moteur de	jan 8	jan 8	1j	1j		0	
2.4	Scène de	jan 9	jan 12	4j	4j		0	
3	▼ Services	jan 15	jan 16	1j 4h	1j 4h		0	
3.1	Chargeur	jan 15	jan 15	1j	1j	4h	0	
3.2	Parseur XI	jan 15	jan 16	1j 4h	1j 4h		0	
4	▼ Données	jan 8	jan 11	3j 1h	3j 1h	3j 3h	0	
4.1	Scène	jan 8	jan 8	1j	1j	4j	0	
4.2	Configurat	jan 9	jan 11	2j 1h	2j 1h	2j 3h	0	
5	Gestion	jan 11	jan 12	1j	1j	2j 3h	0	

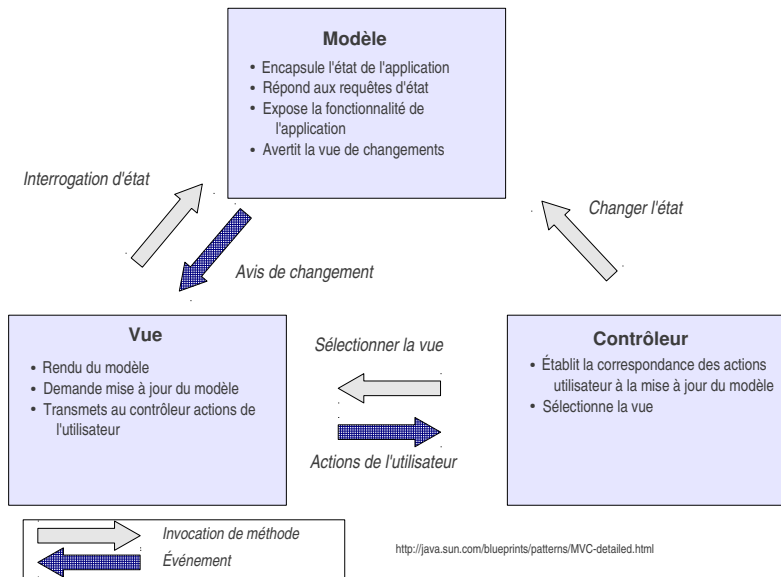
Exemple en gestion de projet

Vue Ressources



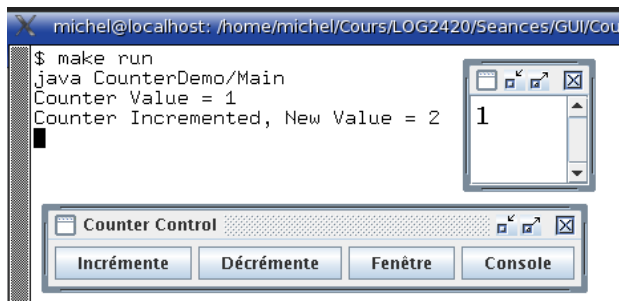
Architecture modèle-vue-contrôleur

Détails



Un exemple en Java – Le “Counter exemple”

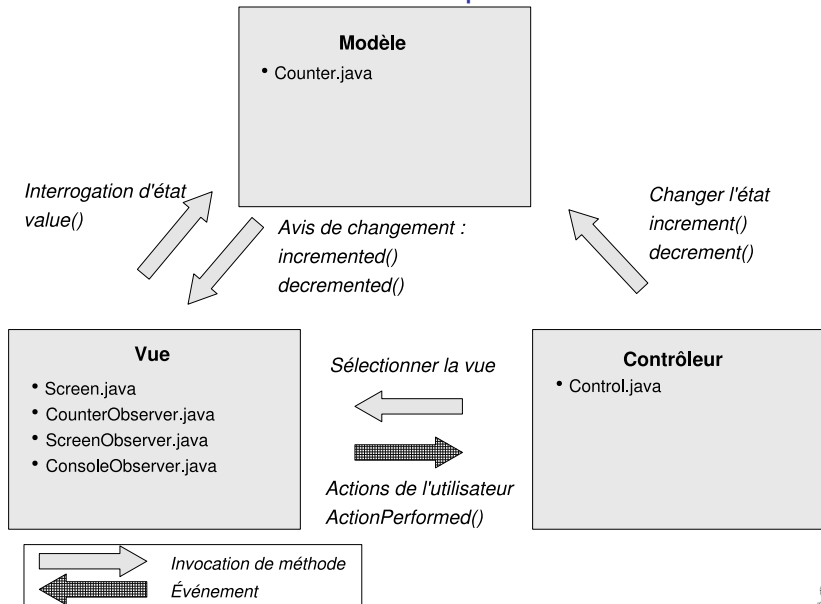
Le “Counter Example”



- ▶ Le programme Java CounterDemo¹ illustre deux vues du modèle du compteur
 - ▶ une vue “console”
 - ▶ une seconde où le compteur s’affiche dans une fenêtre

1. Cet exemple est adapté de <http://courses.csail.mit.edu/6.170/old-www/2006-Fall/lectures/lectures.html>.

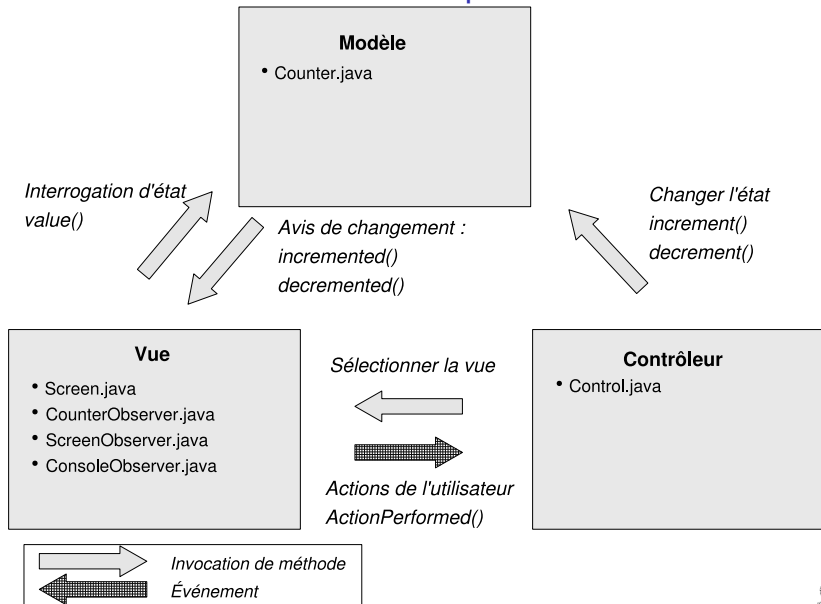
Modèle-vue-contrôleur de l'exemple "Counter demo"



Counter.java

```
public class Counter {  
    int value = 0;  
    CounterObserver observer;  
  
    public void setObserver(CounterObserver co) {  
        observer = co;  
        co.value();  
    }  
    public void increment() {  
        value++;  
        observer.incremented();  
    }  
    public void decrement() {  
        value--;  
        observer.decremented();  
    }  
    public int value() { return value; }  
}
```

Modèle-vue-contrôleur de l'exemple "Counter demo"



Control.java

Initialisation

```
public class Control extends JPanel implements ActionListener {  
    protected JButton b1, b2, b3, b4;  
    CounterObserver screen, console;  
    protected Counter counter;  
    Worker worker;  
  
    public void initialize() {  
        b1 = new JButton("Incrémente");  
        b1.setActionCommand("increment");  
        b1.addActionListener(this);  
        b1.setToolTipText("Incrémente le compteur");  
  
        //Ajouter les composants à ce conteneur en utilisant les  
        FlowLayout par défaut  
        add(b1);  
    }  
  
}
```

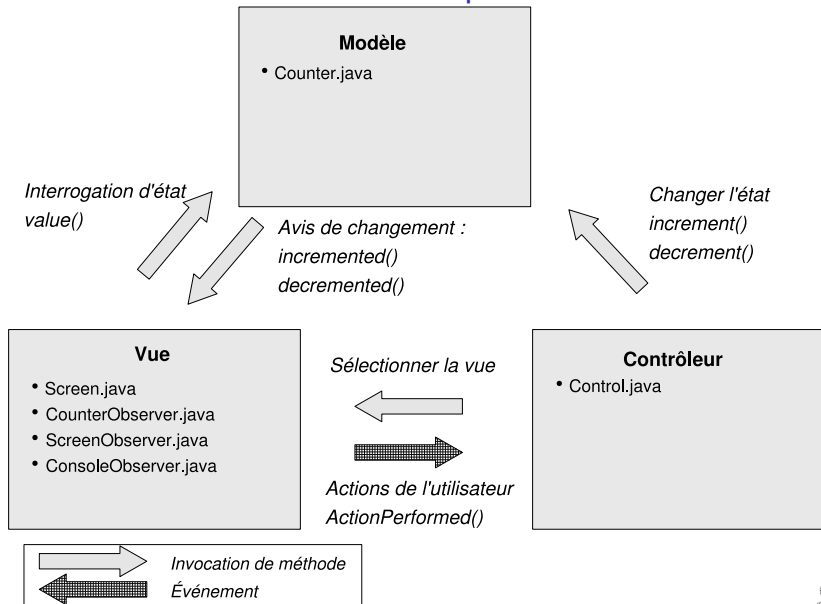
Control.java

Gestion des événements

```
public class Control extends JPanel implements ActionListener {  
    protected JButton b1, b2, b3, b4;  
    CounterObserver screen, console;  
    protected Counter counter;  
    Worker worker;  
    (...)  
    public void actionPerformed(ActionEvent e) {  
        if ("increment".equals(e.getActionCommand())) {  
            counter.increment();  
        } else if ("decrement".equals(e.getActionCommand())) {  
            counter.decrement();  
        } else if ("screen".equals(e.getActionCommand())) {  
            counter.setObserver(screen);  
        } else if ("console".equals(e.getActionCommand())) {  
            counter.setObserver(console);  
        }  
    }  
}
```

(...)

Modèle-vue-contrôleur de l'exemple "Counter demo"



CounterObserver.java

```
interface CounterObserver {  
    public void value();  
    public void incremented();  
    public void decremented();  
}
```


ConsoleObserver.java

```
class ConsoleObserver implements CounterObserver {
    Counter counter;
    ConsoleObserver(Counter c) {
        counter = c;
    }
    public void value() {
        System.out.print("Counter Value = ");
        System.out.print(counter.value());
        System.out.print("\n");
    }
    public void incremented() {
        System.out.print("Counter Incremented, New Value =");
        System.out.print(counter.value());
        System.out.print("\n");
    }
    public void decremented() {
        System.out.print("Counter Decrementated, New Value =");
        System.out.print(counter.value());
        System.out.print("\n");
    }
}
```

ScreenObserver.java

```
class ScreenObserver implements CounterObserver {  
    Screen screen ;  
    Counter counter ;  
    ... /* constructeur */  
    public void value() {  
        String text = Integer.toString(counter.value());  
        screen.setText(text);  
    }  
    public void incremented() {  
        String text = Integer.toString(counter.value());  
        screen.setText(text);  
    }  
    public void decremented() {  
        String text = Integer.toString(counter.value());  
        screen.setText(text);  
    }  
}
```

Tiré d'un article sur l'architecture de Swing :

- ▶ “We quickly discovered that this split didn't work well in practical terms because the view and controller parts of a component required a tight coupling (for example, it was very difficult to write a generic controller that didn't know specifics about the view). So we collapsed these two entities into a single UI (user-interface) object,” (Java Swing Architecture, www.oracle.com/technetwork/java/architecture-142923.html)

La solution appliquée en pratique :

- ▶ Le contrôleur et la vue sont souvent regroupés

Rappel des concepts

- ▶ L'architecture MVC vise à séparer le modèle de la vue et du comportement
- ▶ Le patron “observateur” (*observer*, apparenté au *listener* en Java) est à la base de la modularité et permet de changer de vue dans l'exemple présenté