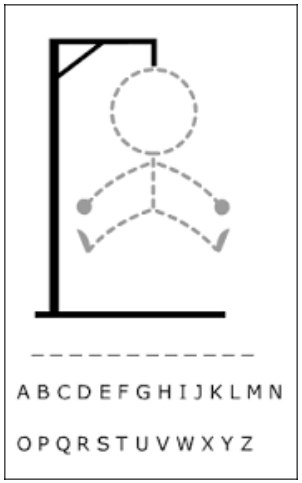
Architecture et Jeux Devint Illustrations

Spécificités DEVINT

- Des jeux simples



- Des jeux avec animation

Spécificités des jeux vidéos

Des éléments spécifiques :

1. une intelligence artificielle
2. les règles du jeu, la gestion de la santé, de l'apparition et l'orchestration des éléments et autres..
3. un moteur qui permet l'orchestration des éléments du jeu un module 2D/3D qui affiche les images à l'écran
4. un module qui gère les sons, la musique selon les événements du jeu
5. un module réseau permettant de faire des jeux multijoueurs ;
6. une interface utilisateur - les menus, l'écran de pause, …

Des patterns pour les Jeux

- Pour les jeux vidéos : Game Programming Patterns

  1. Simplification des rôles MVC avec des contrôleurs intégrés

  http://gameprogrammingpatterns.com/game-loop.html

  2. State Pattern : gestion des états du jeu

  http://gamedev.dreamnoid.com/2009/01/06/game-state-pattern/éléments

  3. MVC Simplifie

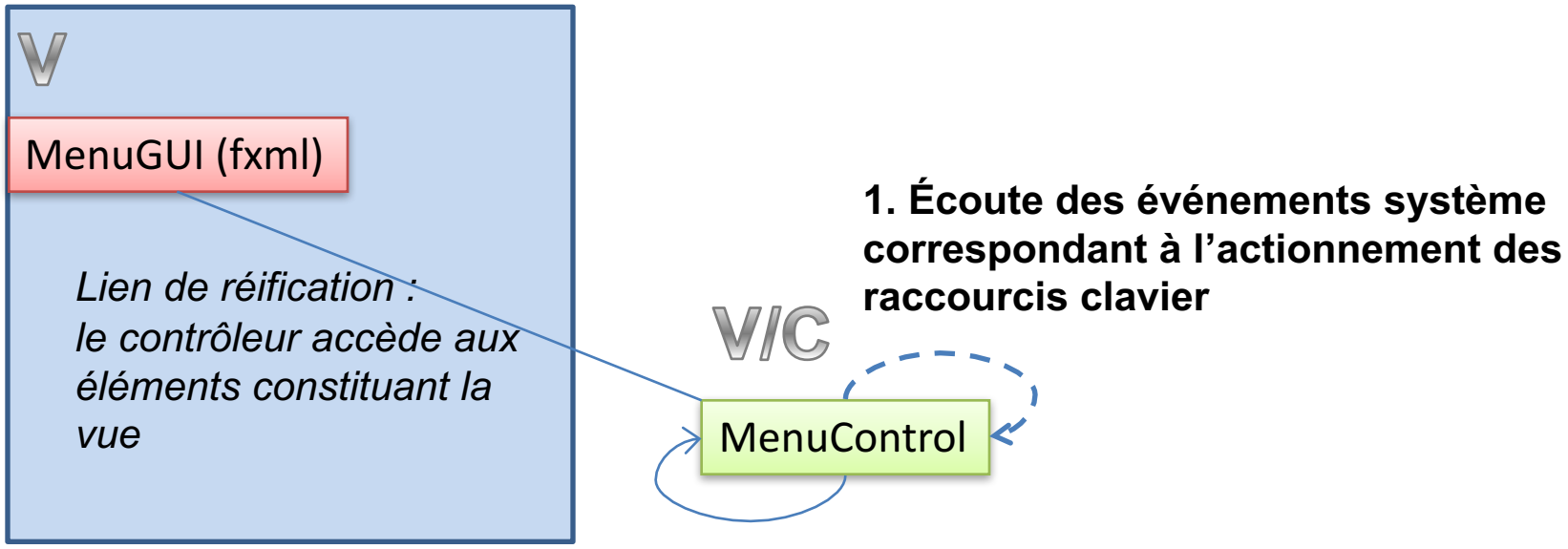
  http://blog.emmanueldeloget.com/index.php?post/2007/06/21/82-quelques-informations-sur-l-architecture-des-jeux-video

Animations et fluidité

- Interactions avec la logique du jeu : gestion des obstacles, etc.
- Rendu du jeu évolution : évolution du paysage

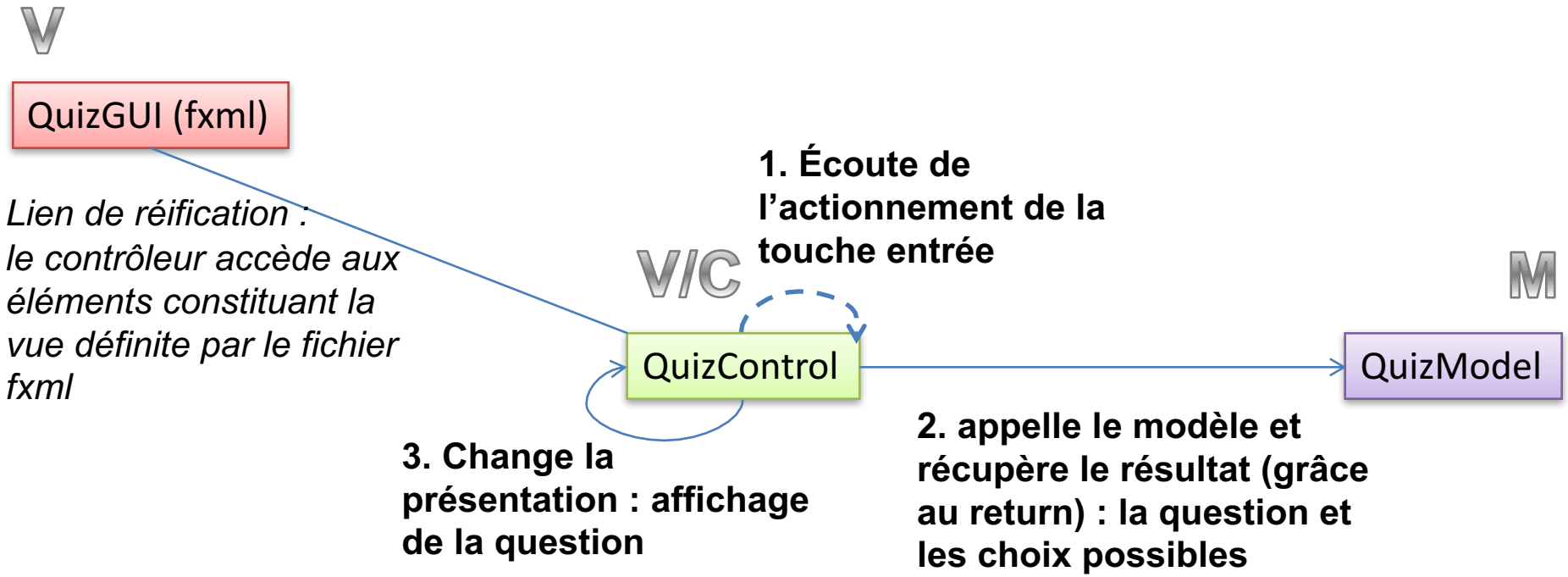
- Game loop
  - loop
    - Model.update
    - Vue.render

Devint – gestion du menu principal



interaction pure

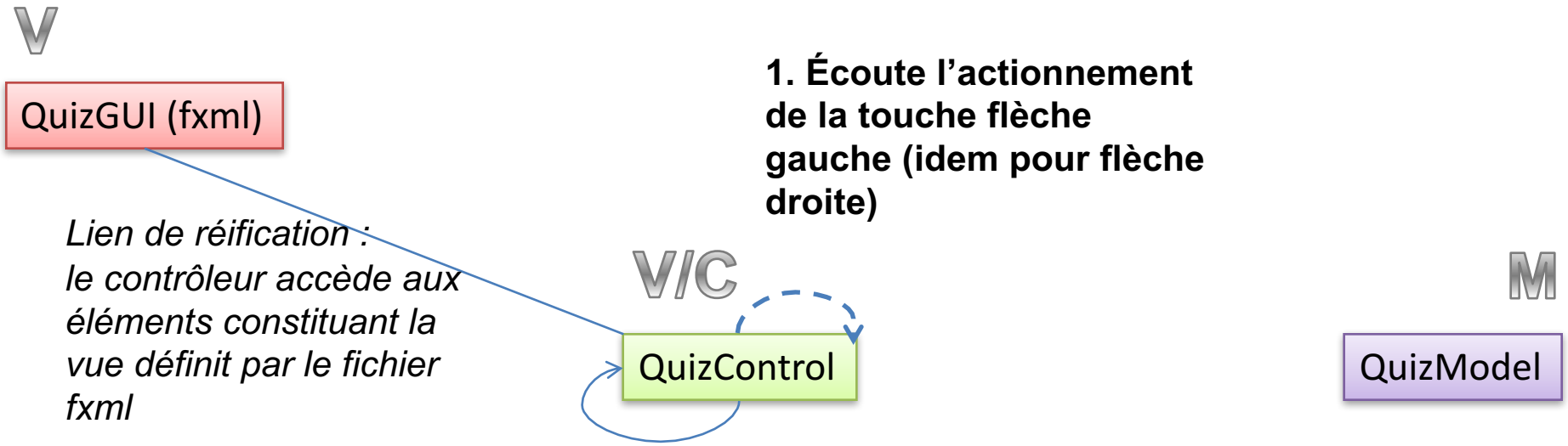
Quiz– On pose la question



Appel au modèle

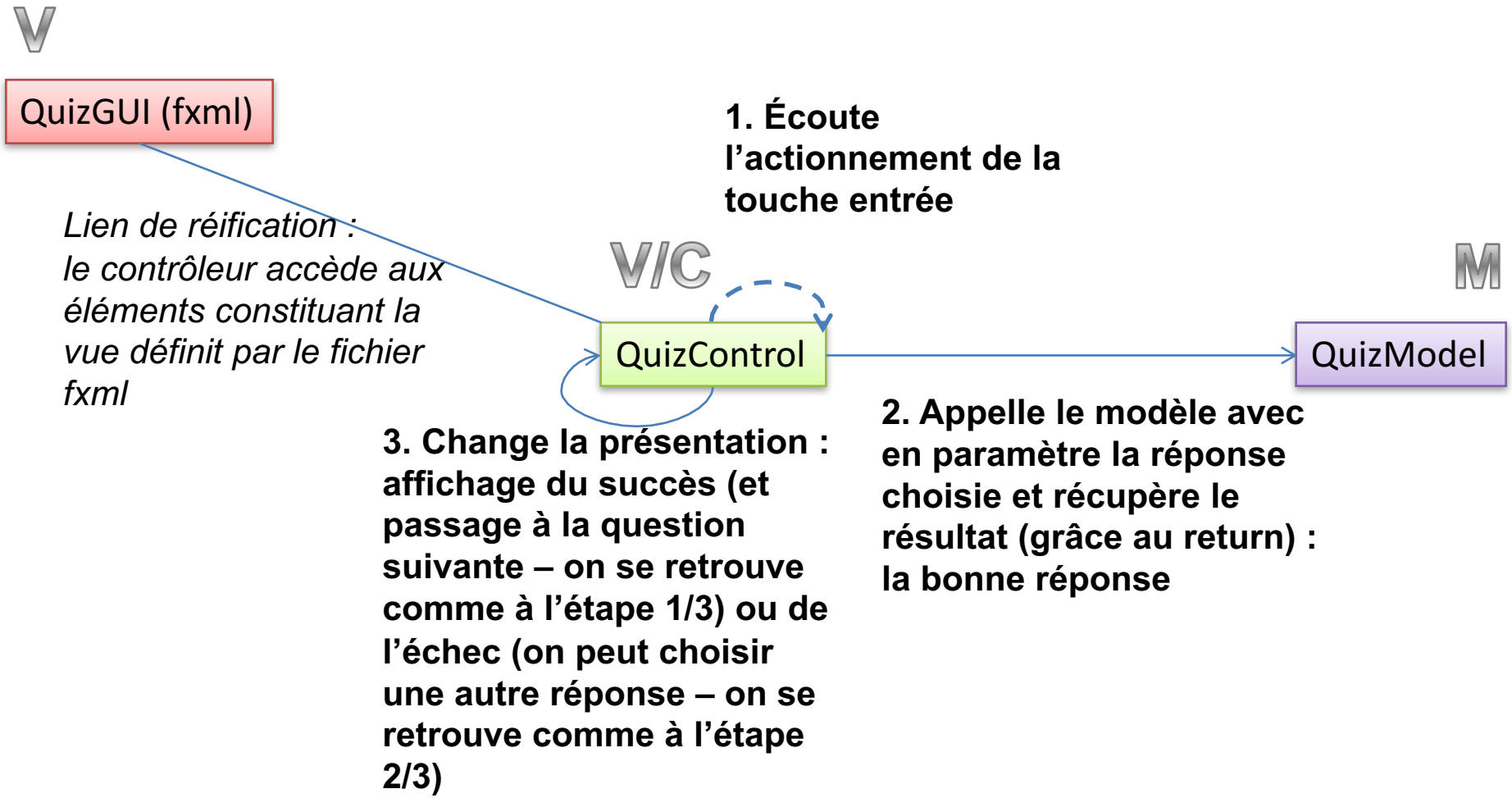
Appel de méthode vs Observer/Observable

Quiz– Déplacement dans les réponses



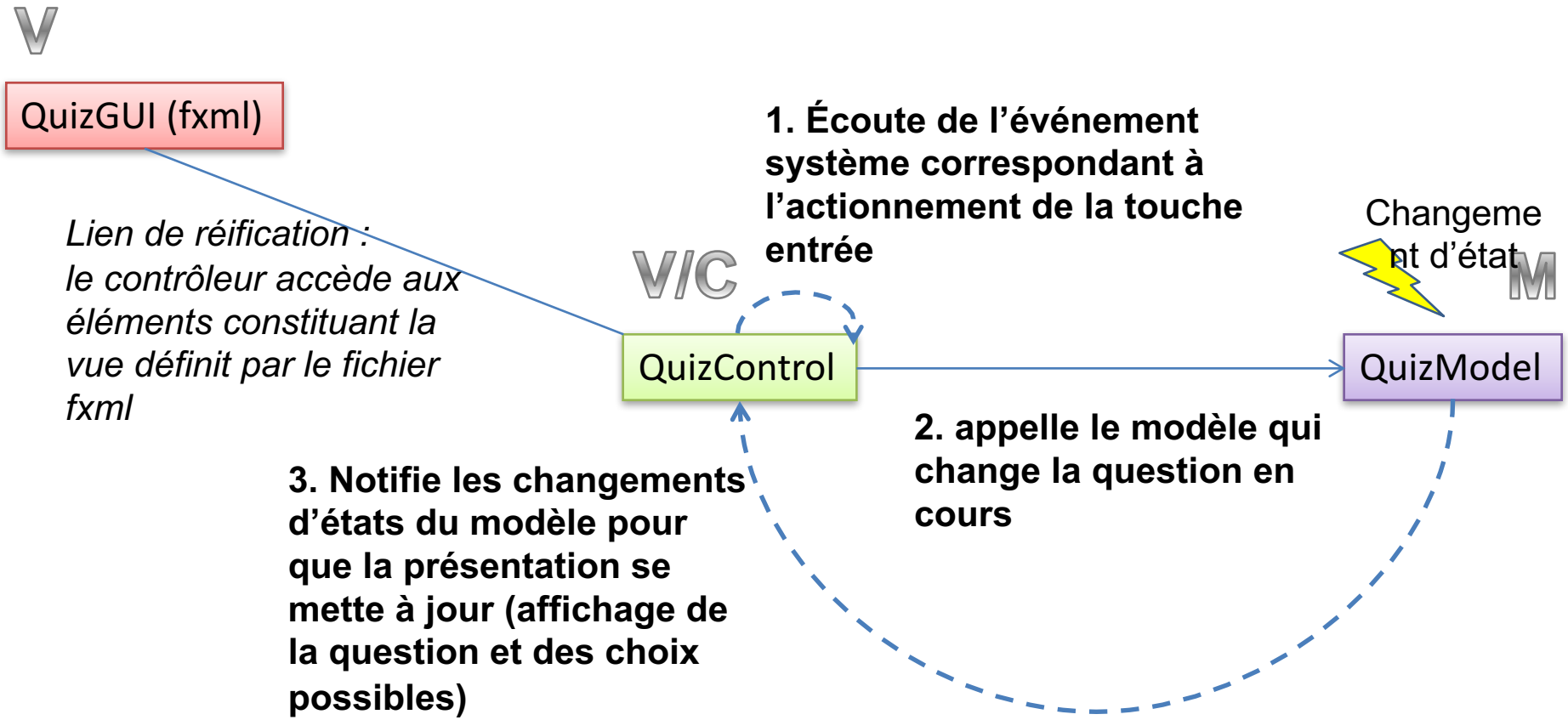
Cas 1 (interaction pure)

Quiz– Vérification de la réponse



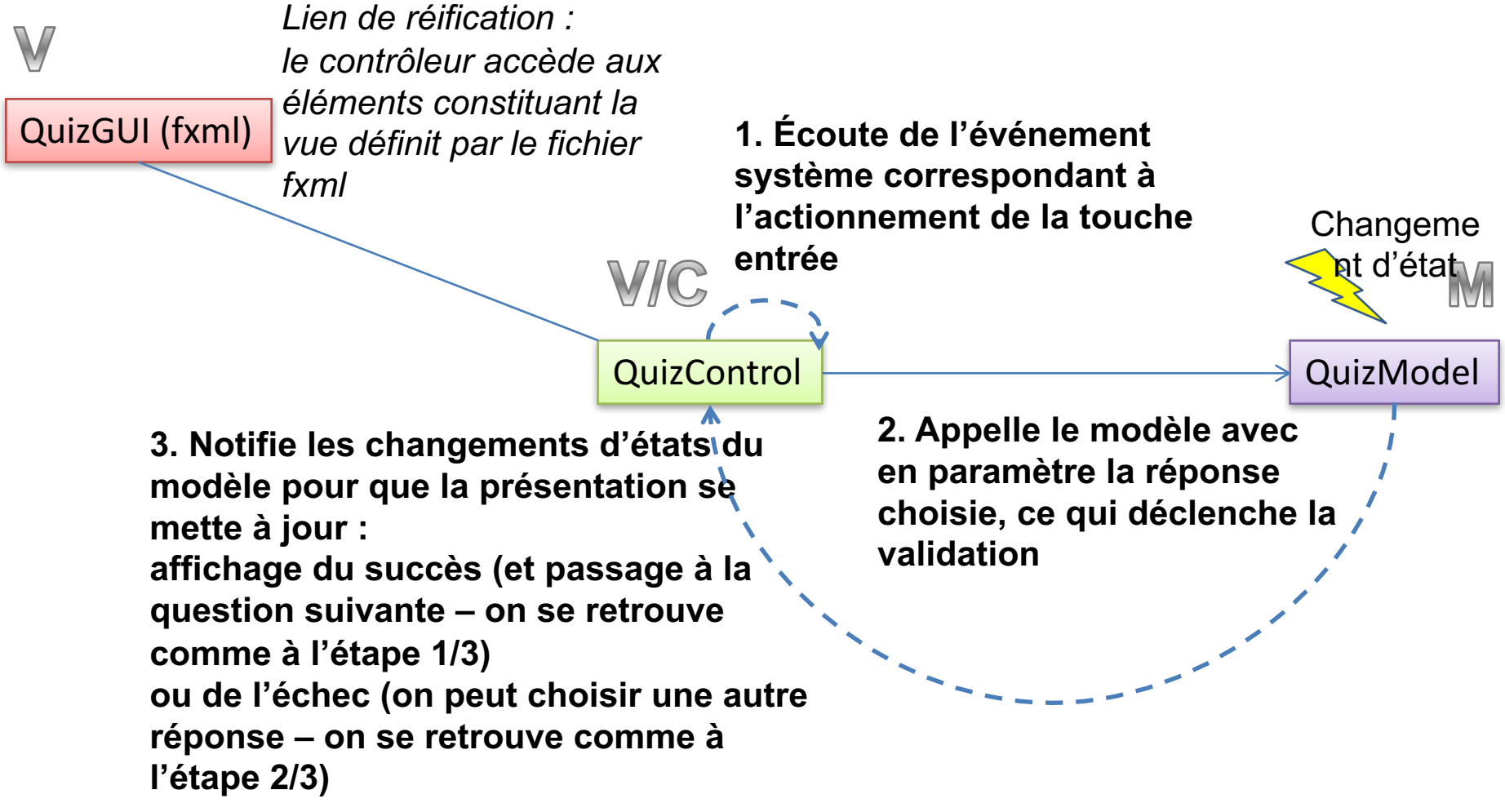
interaction + logique + présentation

Quiz– Variante avec Ecoute du modèle



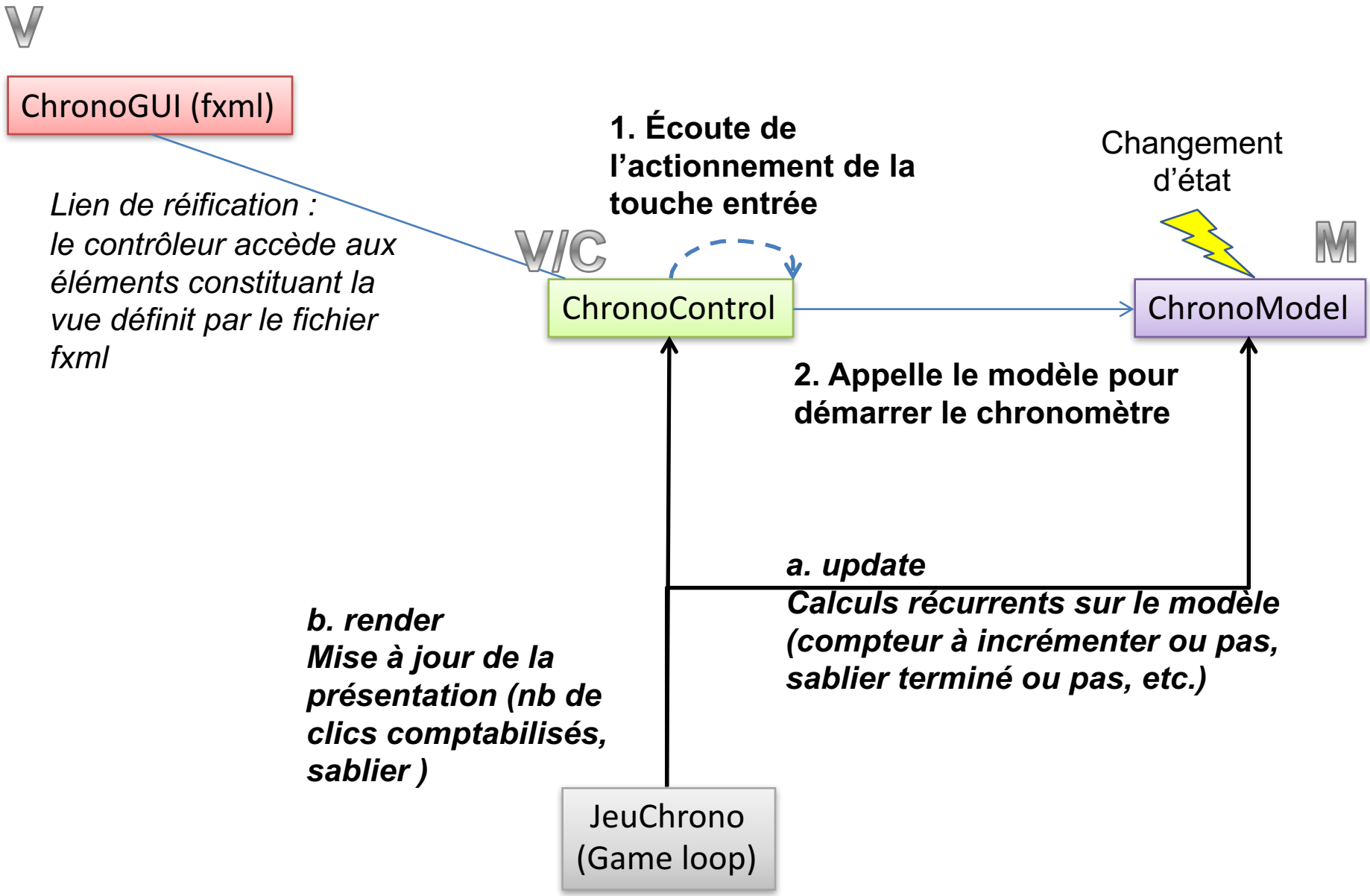
interaction + logique + présentation

Variante avec observation du modèle

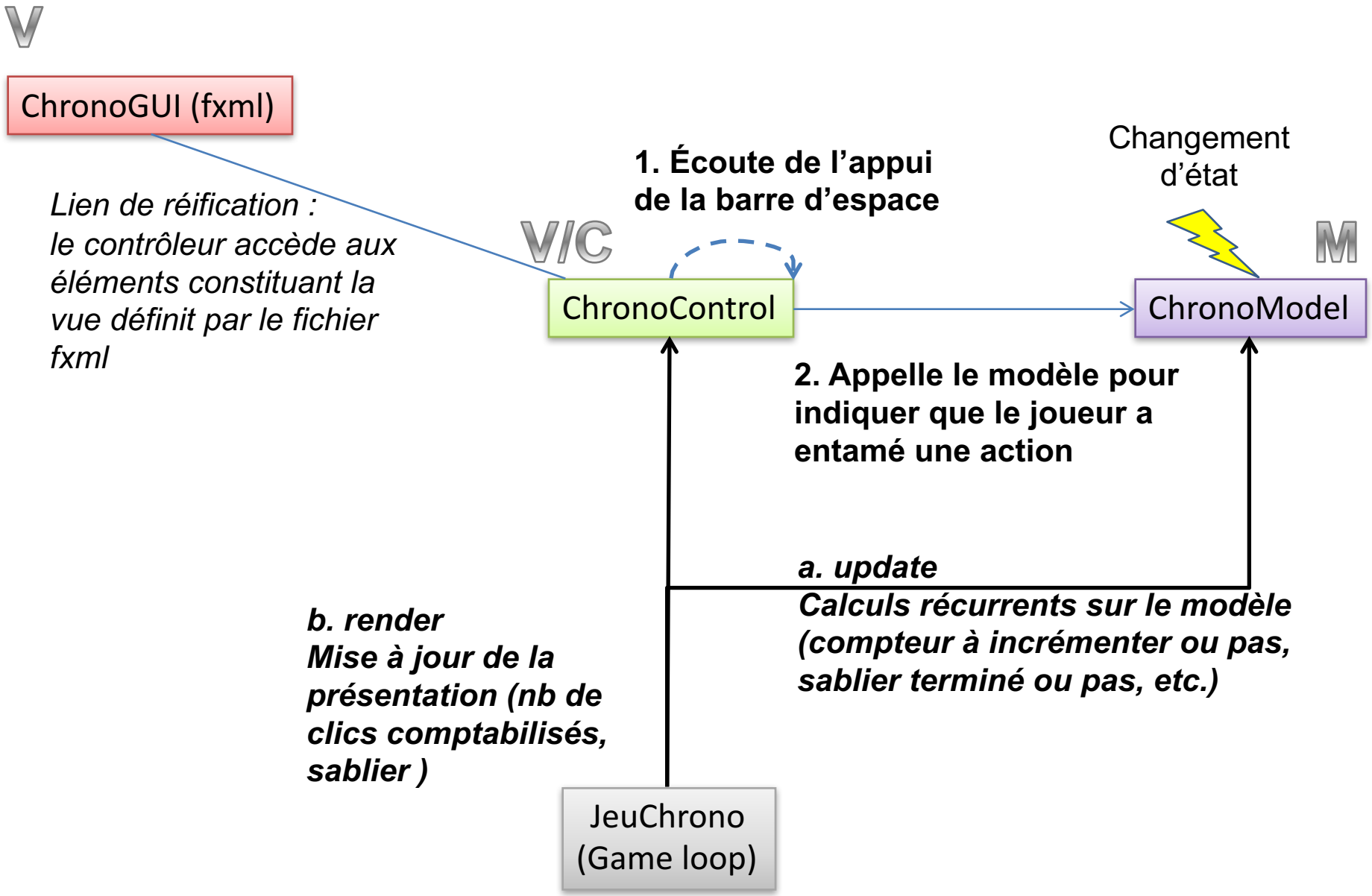


interaction + logique + présentation

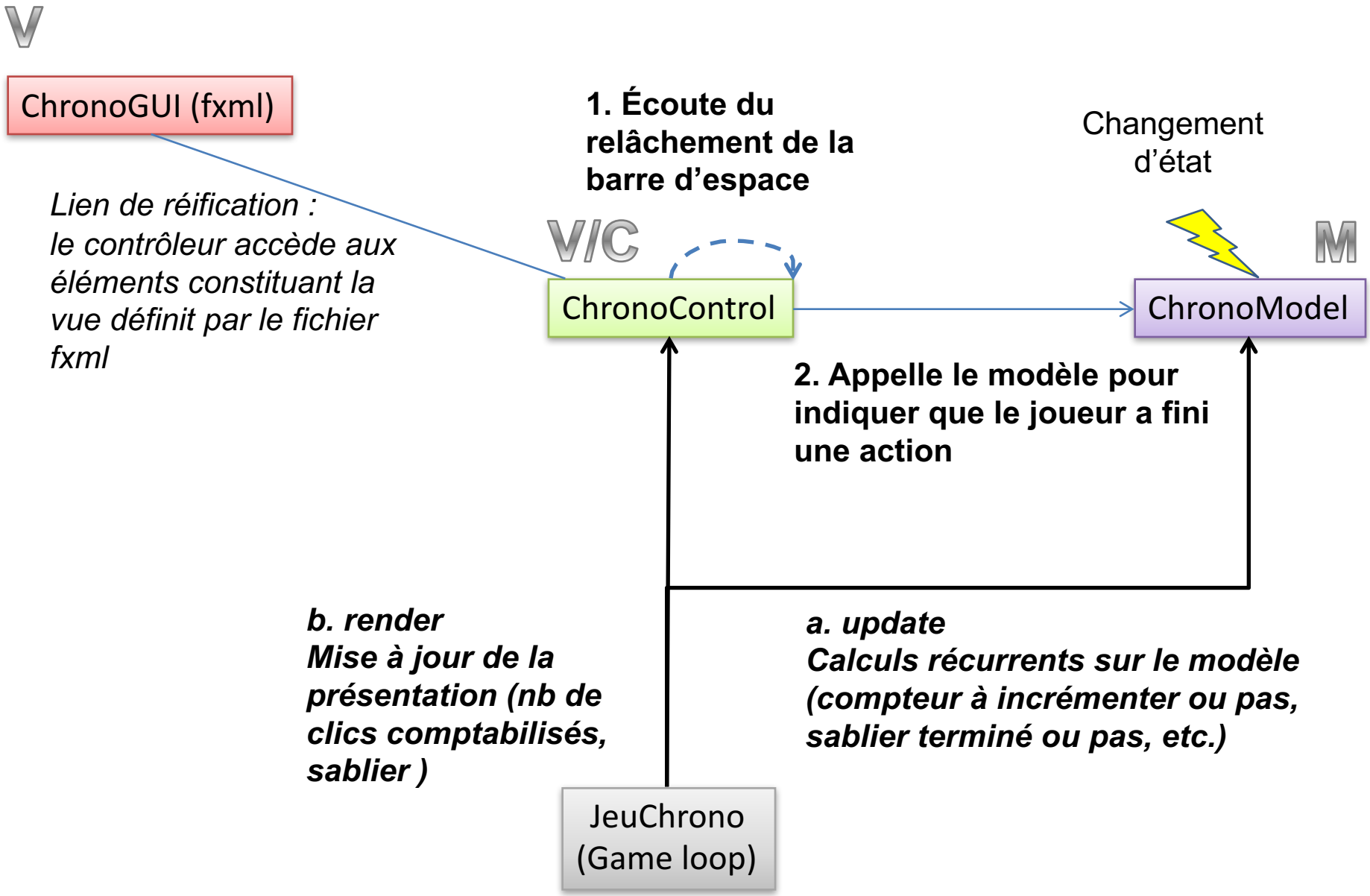
Chrono– Démarrage du chronomètre



Chrono– Suivi de l'interaction « Espace »



Chrono – Suivi de l'interaction « Espace »



Discussions

- VC vs V et C
  - Pourquoi regrouper V et C dans la même entité peut avoir un sens ?
- Le rôle du M
  - L'objet métier représente la logique du jeu
  - quid de conserver le statut des interactions dans le modèle (cf. variantes MVC AM-M VC)
- MVC et GameLoop
  - Bien séparer la Logique du jeu grâce au update de l'évolution du paysage gérée essentiellement par le render

ARCHITECTURE : VARIANTES

Remise en cause du rôle du contrôleur

Objectifs : augmenter la séparation des préoccupations : réactions à l'interactions et la logique de présentation.

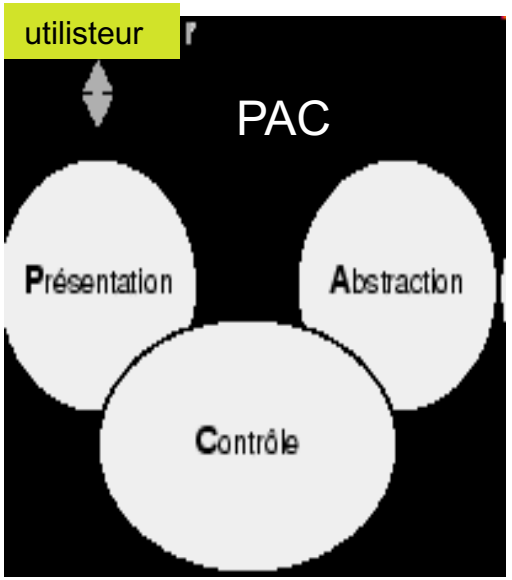
Exemple : si le score est supérieur à 100, la couleur du texte doit devenir verte. La question est donc : qui est responsable de cette logique de présentation ?

Introduction d'autres types de contrôleurs, de présentateurs et ViewModels ou ApplicationModels

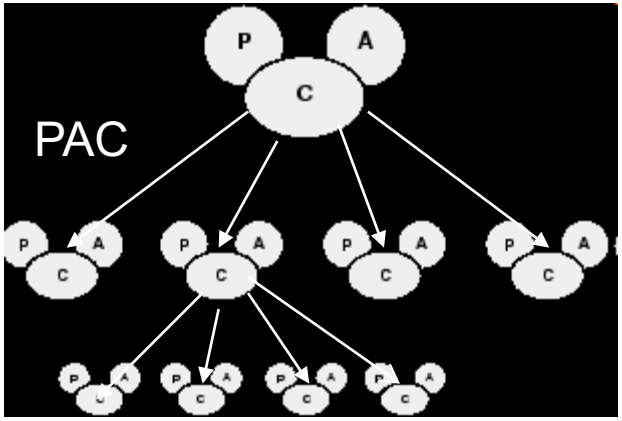
PAC : le contrôleur est central

Modèle à agents **PAC** [Coutaz, 1987]
hiérarchie d'agents explicite

Présentation = structure et comportement en entrée et en sortie de l'agent pour l'utilisateur (fusion des composants vue et contrôleur de MVC)
Abstraction = la partie sémantique de l'agent (équivalent modèle de MVC)
Contrôle = consistance entre la présentation et l'abstraction, navigation dans la hiérarchie
Le composant contrôle n'a pas d'existence explicite dans le modèle MVC.



le modèle PAC peut être appliqué à plusieurs niveaux d'un système interactif. Une application interactive peut ainsi être décrite comme une hiérarchie d'agents disposés sur plusieurs couches d'abstractions

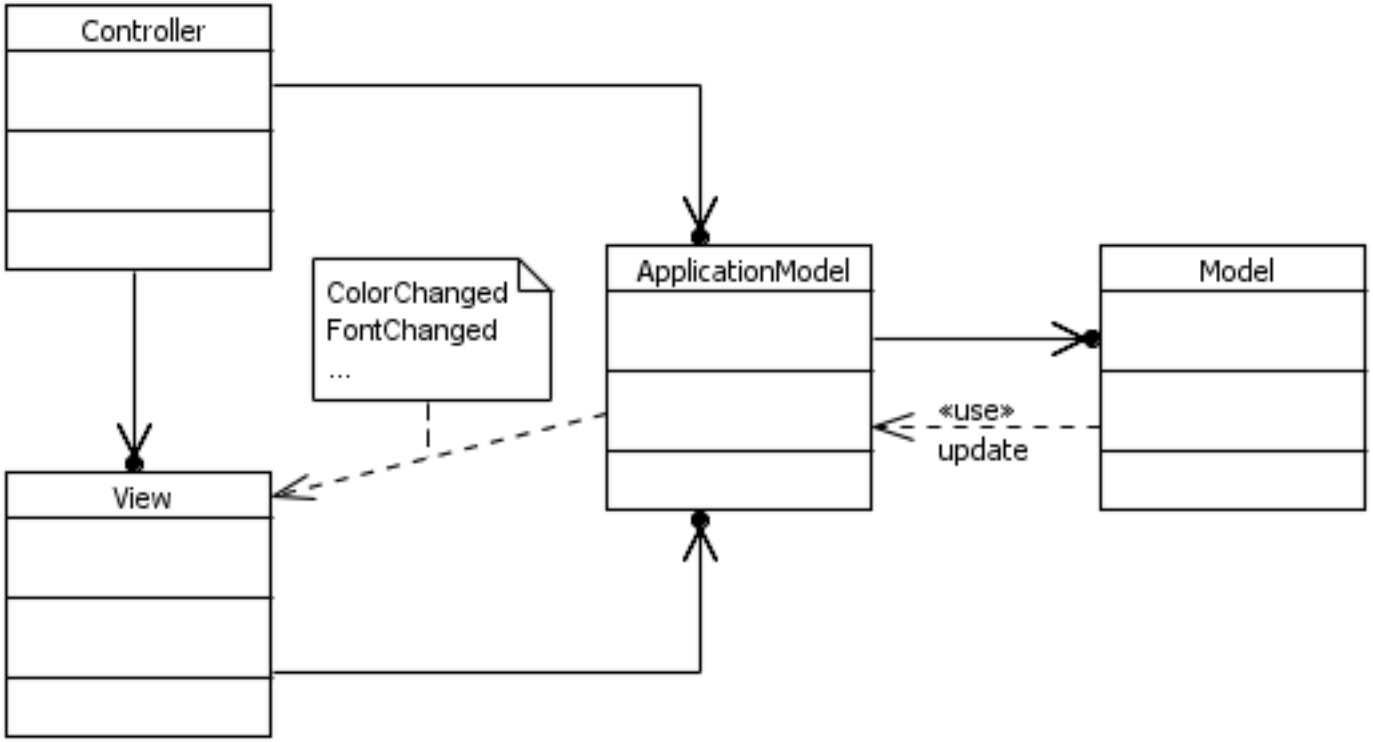


ViewModel ou Application Model

Les responsabilités de logique de présentation sont prise en charge par une entité intermédiaire, nommée « modèle d'application ».

Le modèle d'application agit en intermédiaire entre le modèle et les objets de présentation. Ainsi, le contrôleur n'accède plus directement au modèle, et la vue ne s'enregistre plus comme observateur auprès du modèle, mais ils communiquent avec le modèle d'application et s'enregistrent auprès de lui.

Exemples de variantes

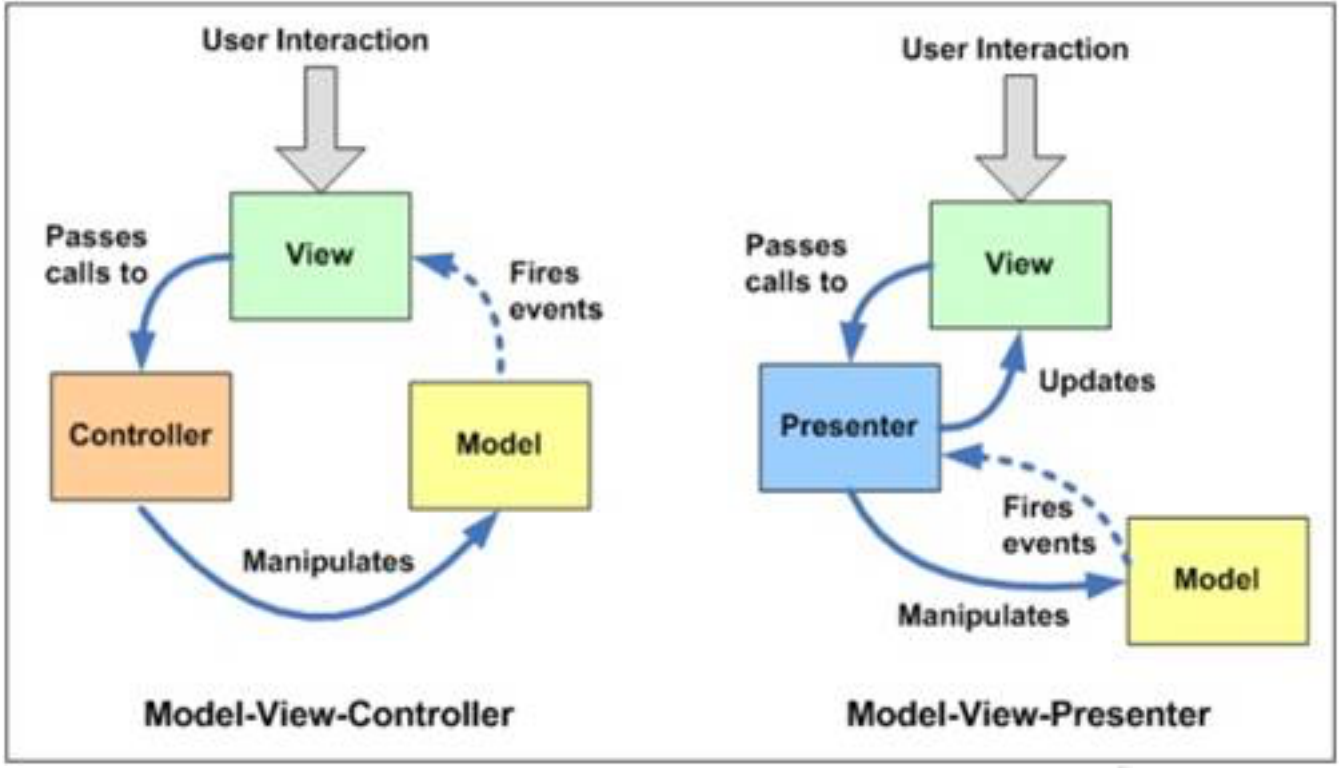


http://cdelmas.developpez.com/articles/conception/variations-autour-du-modele-mvc/

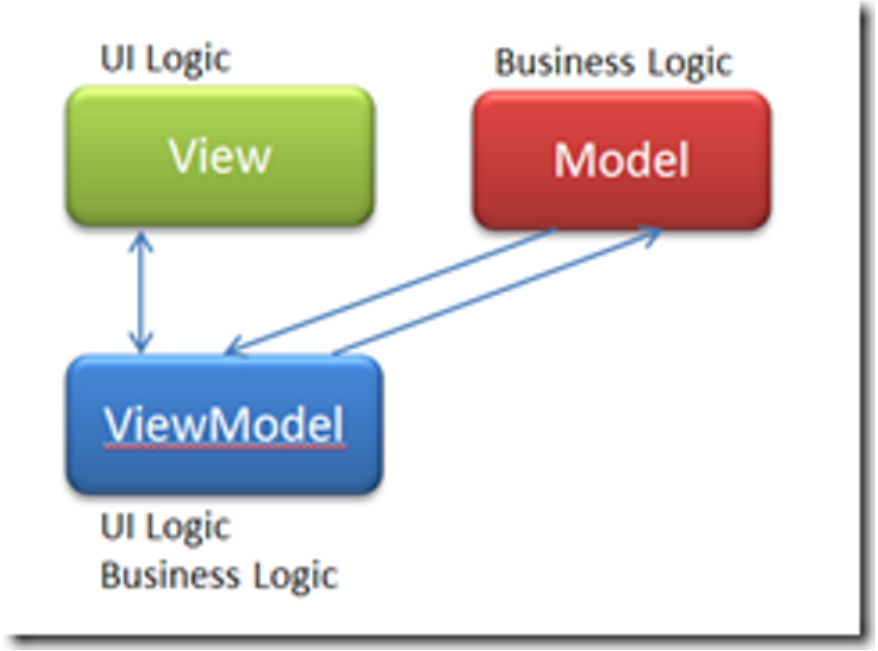
Le controleur devient Présentateur

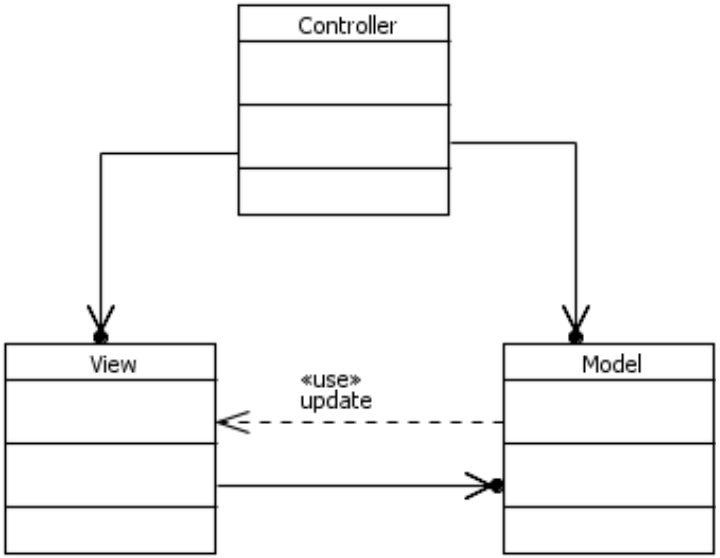
Le présentateur est chargé de la logique de présentation. Il contrôle le modèle, et change la présentation en fonction des règles de l'application. Il est étroitement lié à la vue.

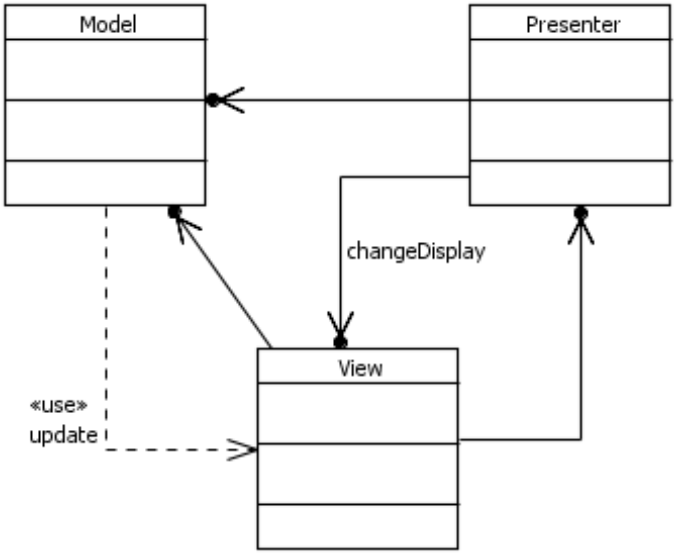
- Deux façons pour communiquer avec la vue : Passive View et Supervising Controler
  - Communication avec la vue : La vue communique directement avec le présentateur en appelant les fonctionnalités à partir d'une instance du présentateur. Alors que le présentateur communique avec la vue à partir d'une interface implémentée par la vue
  - Chaque vue posséde son propre présentateur

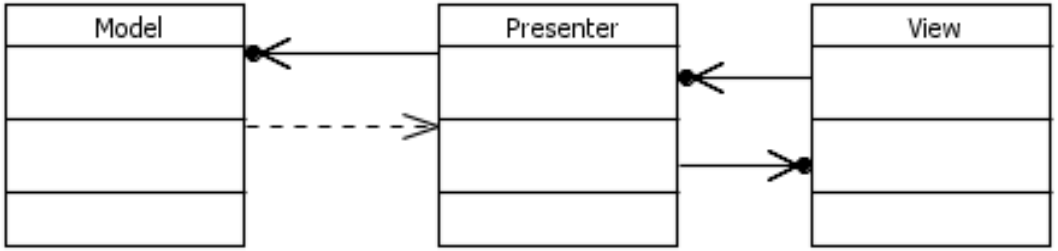

User Interaction
Passes calls to
View
Fires events
Controller
Model
Manipulates
Model-View-Controller
User Interaction
Passes calls to
View
Updates
Presenter
Fires events
Model
Manipulates
Model-View-Presenter

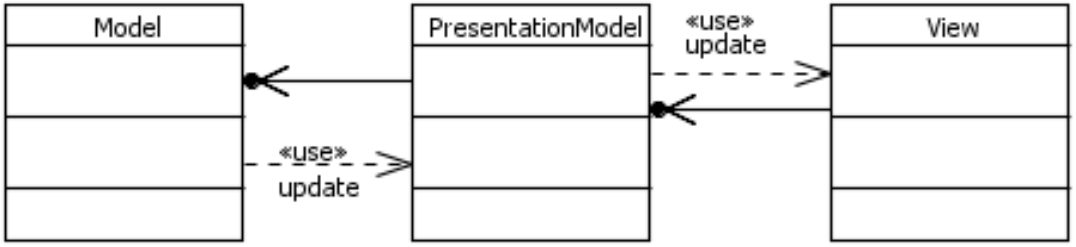
MVP

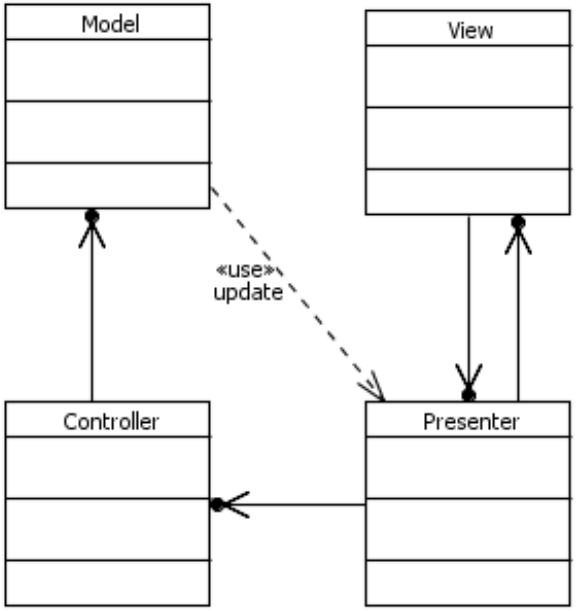

UI Logic
View
Business Logic
Model
ViewModel
UI Logic
Business Logic











http://cdelmas.developpez.com/articles/conception/variations-autour-du-modele-mvc/

Pourquoi autant de modèles ?

De nouveaux modèles inspirés de Arch pour gérer le Device Independence

Des modèles MVC selon les applications :

- quid des applications mobiles
- quid du Web

Que doit on faire ?

Identifier : A quoi sert le modèle ?