

Architecture des IHM

1. Architecture

- Modèle d'architecture
 - Organisation **logicielle** de l'ensemble des fonctionnalités du système
 - *Le modèle du programmeur*
- Rappel: ne pas confondre avec le Modèle conceptuel
 - Organisation **conceptuelle** de l'ensemble de fonctionnalités du système en un ensemble des commandes et de leurs effets
 - *Modèle de l'utilisateur*
- Remarque
 - Du modèle d'architecture découlent ...
 - ... les performances/maintenabilité/modularité du système
 - Du modèle conceptuel découlent ...
 - ... les performances, la satisfaction et l'état de l'utilisateur

Modèle d'architecture logicielle

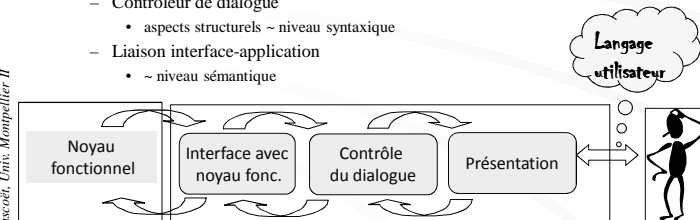
- Modèle de base
 - Se retrouve dans tous les modèles
 - Séparation de l'interface et de l'application



- Autres modèles d'architecture
 - Modèle Seeheim ou modèle langage
 - Modèle objet
 - MVC
 - Modèle multi-agents: PAC

Modèle Seeheim : modèle séminal

- Modèle langage
 - Composant présentation
 - aspects externes de l'interaction ~ niveau lexical
 - Contrôleur de dialogue
 - aspects structurels ~ niveau syntaxique
 - Liaison interface-application
 - ~ niveau sémantique



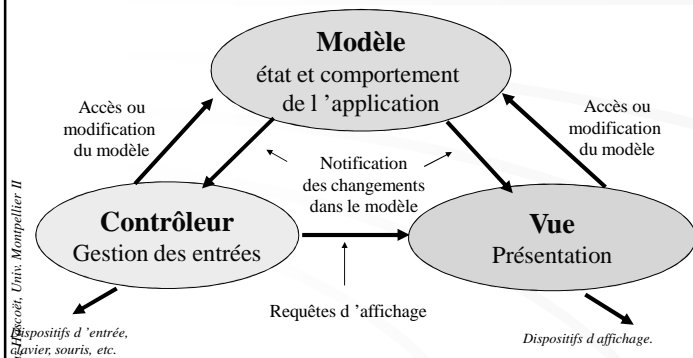
Modèles Objets

- Modèle général intrinsèque des IHM à manipulation directe
- Existe dans de multiples variantes
 - comme modèle architectural
 - comme modèle conceptuel
 - comme modèle conceptuel et architectural
- Engendre la construction de modèles plus élaborés
 - MVC
 - PAC
 - ...

Modèle-Vue-Contrôleur

- Origine de MVC
 - Modèle issu de Smalltalk-80 Interactive Programming Environment
 - Adopté par de nombreuses boîtes à outils graphiques
 - Smalltalk-80, Java AWT 1.2, Microsoft Foundation Class Library, X-Designer, etc.
- Principe de base
 - M : Modèle
 - représente les données de l'application (implémente fonctionnalités et structures de données)
 - V: Vue
 - présente l'information aux utilisateurs en utilisant les données issues des modèles
 - C: Contrôleur
 - se charge de l'interaction avec l'utilisateur

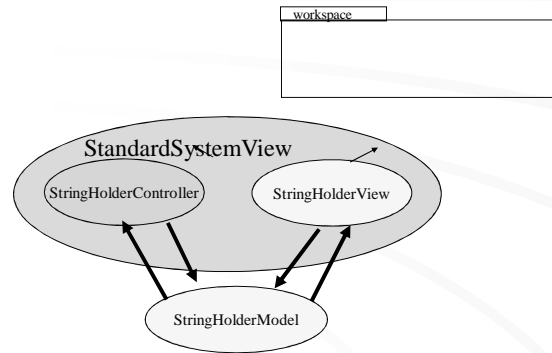
Communications entre les composants MVC



Modèle multi-vues

- Modèle asymétrique
 - Une paire Contrôleur/Vue est associée à un seul Modèle
 - Un modèle peut se voir associé plusieurs paires Contrôleur/Vue
- Listes de dépendants et notification
 - Les paires Vue/Contrôleurs d'un modèle sont enregistrées au niveau de ce modèle dans sa liste de « dépendants »
 - Lorsque le modèle change, tous les dépendants sont notifiés
- Cycle standard d'interaction dans MVC
 - Interaction utilisateur => activation d'un contrôleur => modification du modèle => notification et modification des vues

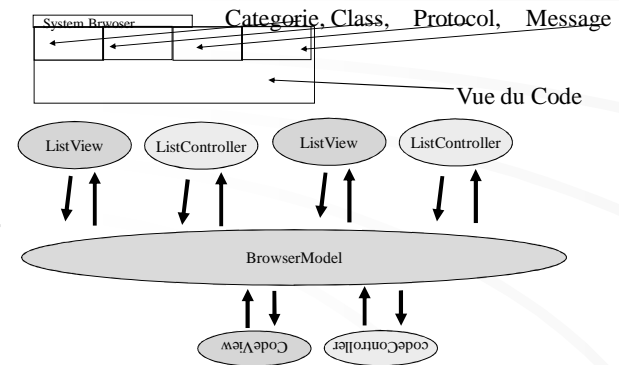
Exemple : Workspace



Montauz Hascoët, Univ. Montpellier II

9

Exemple : Browser



Montauz Hascoët, Univ. Montpellier II

10

Conclusion sur MVC

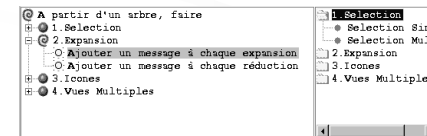
- Avantages
 - vues multiples synchronisées
 - vues et contrôleurs modulaires
 - développement de composants ré-utilisables
 - propagation naturelle du « look and feel »
 - cohérence interne et externe des interfaces
- Inconvénients
 - complexité de communication entre les composants

Montauz Hascoët, Univ. Montpellier II

11

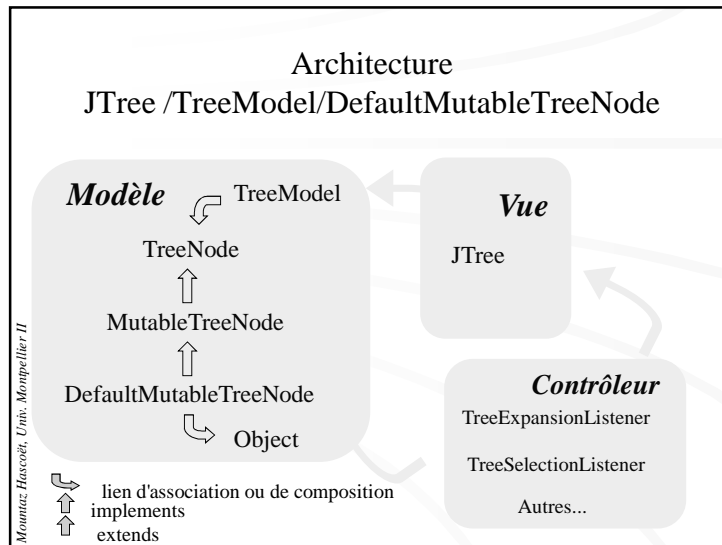
Le composant Arbre de Swing

- Une application de MVC
- class Jtree
 - C'est la vue de l'arbre
 - => ne contient pas les données
 - Composée de nœuds qui font référence aux nœuds qui contiennent les données



Montauz Hascoët, Univ. Montpellier II

12



Modèle: interface TreeModel

- Les méthodes pour la liaison avec le Contrôleur


```
void addTreeModelListener(TreeModelListener l)
void
  removeTreeModelListener(TreeModelListener l)
  • Ajout/suppression d'un contrôleur
```
- Quelques méthodes rudimentaires de parcours d'arbre


```
Object getChild(Object parent, int index)
int getChildCount(Object parent)
int getIndexOfChild(Object parent,
Object child)
Object getRoot()
boolean isLeaf(Object node)
```

Mounitz Hascoët, Univ. Montpellier II

14

Modèle: interface TreeNode

Des méthodes d'accès/ lecture de base

```
Enumeration children()
  Retourne les enfants du nœud (sous la forme d'une
  Enumeration).
TreeNode getParent()
  rend le père du nœud appelant

TreeNode getChildAt(int childIndex)
  rend le ième enfant du nœud appelant.
int getChildCount()
  nb d'enfants du nœud appelant.
int getIndex(TreeNode node)
  rang du nœud node parmi les enfants de nœud
  appelant.
boolean isLeaf()
  Vrai si le nœud appelant est une feuille
```

Mounitz Hascoët, Univ. Montpellier II

15

Implémentation par défaut d'un TreeNode : DefaultMutableTreeNode

Toutes les méthodes de parcours/accès/manipulation d'arbre

```
void add(MutableTreeNode newChild)
  ajoute newChild à la fin des enfants du nœud appelant

Enumeration children()
  crée et retourne une Enumeration pour parcourir les enfants du nœud appelant

Enumeration breadthFirstEnumeration([Enumeration depthFirstEnumeration])
  crée et retourne une Enumeration qui parcourt le ss-arbre du nœud appelant en
  largeur [en profondeur]
int getDepth()
  retourne la profondeur du [sous]-arbre du nœud appelant
TreeNode getFirstChild() [TreeNode getLastChild()]
  le premier enfant du nœud appelant
```

Mounitz Hascoët, Univ. Montpellier II

16

DefaultMutableTreeNode (suite)

```
int getLeafCount()
    nb de feuilles qui sont des descendantes du nœud appelant

DefaultMutableTreeNode getFirstLeaf()
    [DefaultMutableTreeNode getLastLeaf()]
    le nœud lui-même si c'est une feuille ou la 1ère feuille descendant du
    nœud

DefaultMutableTreeNode getNextSibling()
    premier frère du nœud appelant

TreeNode[] getPath()
    le tableau des nœuds qui conduisent à ce nœud depuis la racine

Object getUserObject()[void
    setUserObject(Object userObject)]
    le contenu du nœud - c'est ici que les données se trouvent

String toString()
    retourne le résultat de la méthode toString de l'objet contenu dans le
    nœud
...

```

Montaniz Hascoët, Univ. Montpellier II

17

Vue: la classe JTree

- De nombreux constructeurs
 - JTree()
 - JTree(TreeModel newModel)
 - JTree(TreeNode root)
- Liaison avec le contrôleur
 - void addTreeExpansionListener(TreeExpansionListener tel)
 - void addTreeSelectionListener(TreeSelectionListener tsl)
- Liaison avec le modèle
 - TreeModel getModel()
 - void setModel(TreeModel newModel)
- Toutes les méthodes pour l'affichage des nœuds
 - setRootVisible() •scrollPathToVisible() •scrollRowToVisible() ...

Montaniz Hascoët, Univ. Montpellier II

18

Contrôleurs: à écrire à partir des GE

- Expansion de l'arbre
 - TreeExpansionListener
 - void treeCollapsed(TreeExpansionEvent event)
 - void treeExpanded(TreeExpansionEvent event)
 - TreeExpansionEvent
 - TreePath getPath() (si TreePath p, p.getPath())
 - retourne le tableau des nœuds du path
- Selection des items dans l'arbre
 - TreeSelectionListener
 - void valueChanged(TreeSelectionEvent e)
 - TreeSelectionEvent
 - TreePath getPath()
- Divers
 - pour des interactions particulières on peut utiliser les ge non dédiés.

Montaniz Hascoët, Univ. Montpellier II

19