interaction homme-machine

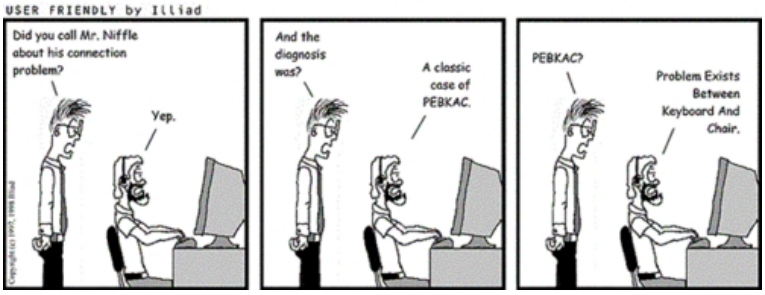
introduction, conception logicielle, outils

eric lecolinet
telecom paristech
www.telecom-paristech.fr/~elc

INTERACTION HOMME-MACHINE

IHM (HCI)

- Conception
- Implémentation
- Evaluation
  - de systèmes **interactifs**
  - destinés à des **humains**



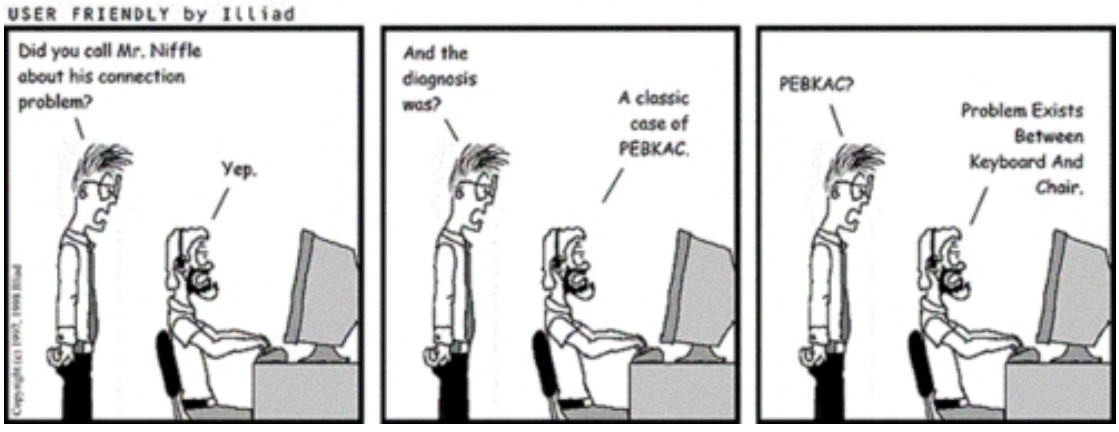
THE PEBKAC PROBLEM...

IHM

Conception, implémentation, évaluation

- de systèmes **interactifs**
- destinés à des **humains**

THE PEBKAC PROBLEM
Problem
Exists
Between
Keyboard
And
Chair



IHM

Pluridisciplinarité

- **Ingénierie**
  - informatique, génie logiciel, électronique ...
- **Facteurs humains**
  - ergonomie, psychologie cognitive, sociologie ...
- **Design**
  - design industriel, arts graphiques, typographie

Processus de développement adapté

- méthodes de conception / réalisation / évaluation appropriées

Enjeux (1)

1. Les IHMs sont omniprésentes et incontournables

- **Elément de productivité** essentiel :
  - coût d'apprentissage
  - exploitation des fonctionnalités avancées
  - facilité / rapidité d'exécution
  - fatigue
  - risques d'erreur
  - évolutions futures, maintenance ...

La catastrophe du Mont Saint Odile

Crash d'un Airbus A-320 le 20 janvier 1992

- 87 morts, 9 survivants
- Mise en examen en janvier 1997 pour "homicide et blessures involontaires" en relation avec **« l'ergonomie du mode de descente de l'appareil »**

Wikipedia

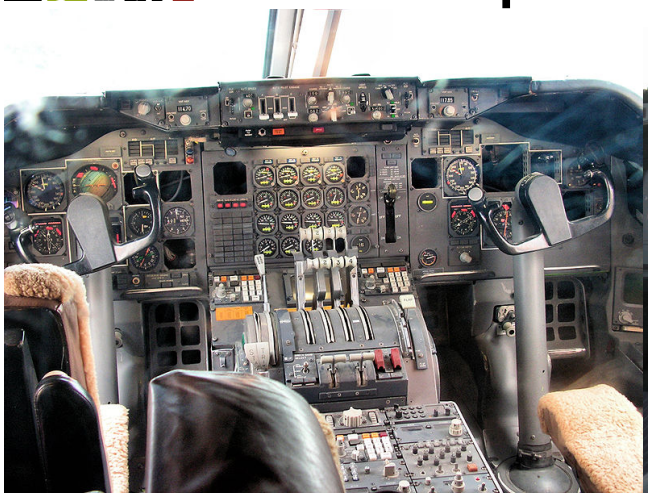
- Une confusion liée à **l'affichage peu différencié** de deux valeurs très différentes selon le mode de descente sélectionné :
  - **angle de descente** (mode FPA - Flight Path Angle)
  - ou **vitesse verticale** (mode VS - Vertical Speed)

La catastrophe du Mont Saint Odile

Wikipedia (...)

- Le pilote a enregistré sur l'ordinateur de bord la valeur **33** croyant être en mode FPA alors qu'il était en mode VS
- Au lieu de programmer un **angle** de descente de **3,3°** il a en fait programmé une **vitesse** de descente de **3 300** pieds/mn (16.7 m/s)
- L'avion opère alors une descente trop rapide à un taux de **quatre fois supérieur** au taux normal (3 300 pieds/minute au lieu de 800–900 pieds/ minute (4 à 4.5 m/s))

La catastrophe du Mont Saint Odile



Erreurs d'ergonomie

- mauvaise visibilité des modes, pas d'unités, pas de repère visuel (3.3 vs. 3300)

Mais aussi un nouveau type de cockpit, différent de l'existant

- à gauche un Boeing 747 (ancien modèle), à droite un Airbus A330

Moins grave mais symptomatique

- « **OK** vs. **DoIt** » sur les boîtes de dialogue

- Aujourd'hui, informaticien sur une hotline, je reçois un appel : "Monsieur, juste pour être sûr, si je mets la mouette dans la case, ça veut bien dire que je n'aurai pas à retaper mon mot de passe ?" Une minute de réflexion... Ah OK... Mouette dans la case = cocher la case. VDM

- Aujourd'hui, on m'a présenté ma nouvelle assistante, Brigitte, qui accuse déjà 35 ans de boutique. Pour inscrire dans Word un nombre venant d'Excel, Denise imprime son document Excel, ferme Excel, ouvre Word et recopie scrupuleusement la feuille papier qu'elle a plaquée contre son écran. VDM

Enjeux (2)

1. Les IHMs sont omniprésentes et incontournables

- élément de productivité essentiel ...

2. Le coût de développement des IHMs est élevé

- souvent 50% (ou plus) du code / du travail total !

➔ Double objectif

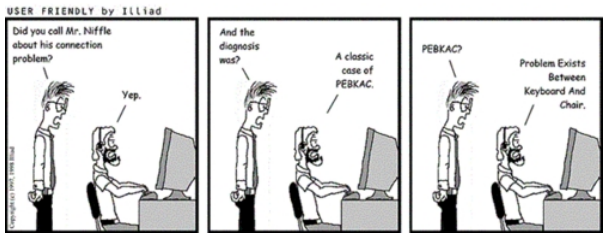
- concevoir des systèmes **fiables** et **faciles à utiliser**
- en un **temps** et à un **coût** raisonnables ...

  ➔ optimiser le ratio performances / coût

Approche centrée utilisateur

La science, la technique s'adaptent à l'utilisateur ...

- et non l'inverse !
- informatiser au lieu d'automatiser



Facteurs humains

- difficiles à intégrer

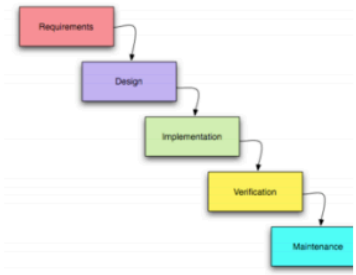
➔ méthodologies de conception

Concept d'utilisabilité (usability)

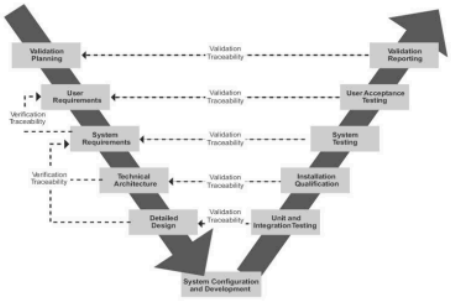
- facilité d'apprentissage et d'utilisation

Modèles de conception

- Modèles de conception : Waterfall (modèle en cascade), V, Spiral …
  - Souvent centrés systèmes
  - L'utilisateur n'intervient qu'en amont et en aval de la conception (analyse et évaluation)



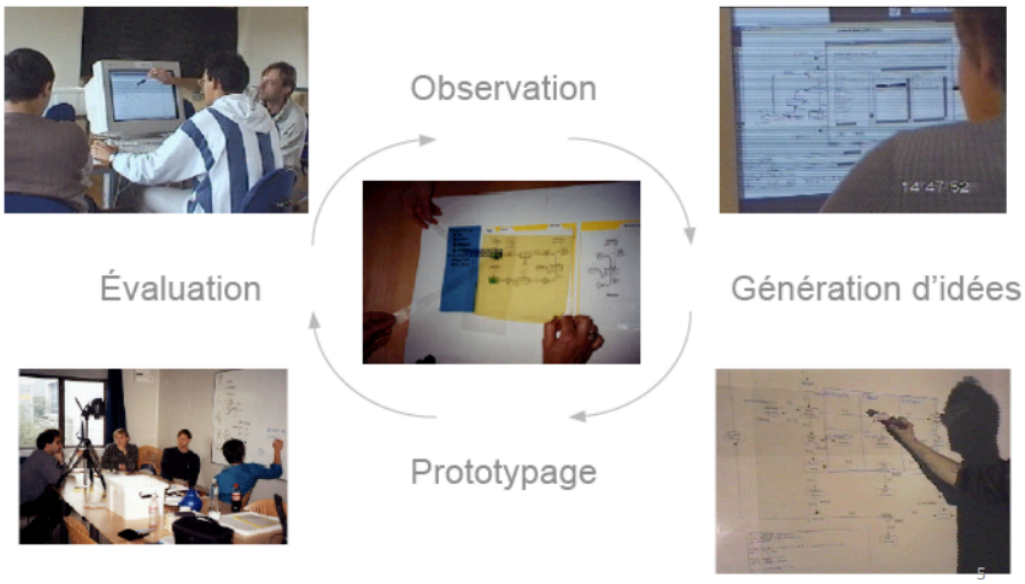
Waterfall



V

Spiral

Conception participative (et itérative)



- réalisation rapide de prototypes ou de « mock ups »
- approche empirique (essais-erreurs): aspect chaotique de la conception...

Exemple: Planning Word 6

jour	tâche prévue
vendredi	vérification de l'état des features à tester
lundi	développement des tâches de test de la semaine
mardi	reception du prototype. pre-test en interne du proto pour en déterminer les limites
mercredi	test avec sujets les responsables développeurs participent à l'obsercation
jeudi / ven	analyse des tests, recommandations, synthèses

- sur plusieurs mois
- 43 tests utilisateurs / plus de 250 participants

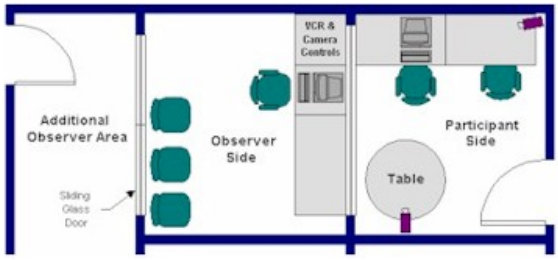
Conception participative

Utilisateurs

- experts du domaine concerné
- prennent un rôle actif dans la conception

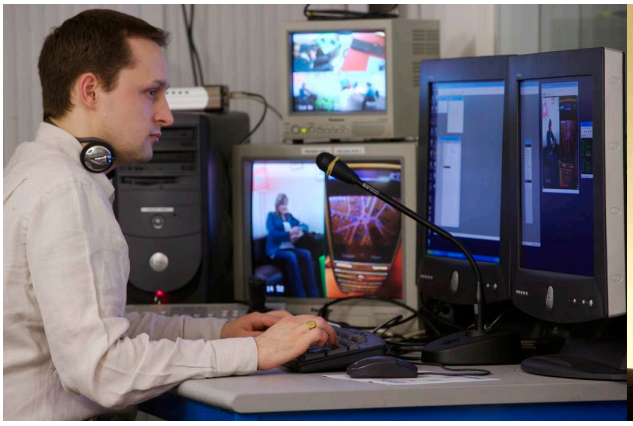
Importance des facteurs humains

- Faire « entrer l'humain dans la boucle »



Usability Lab (LUTIN à La Villette)

http://www.lutin-userlab.fr/

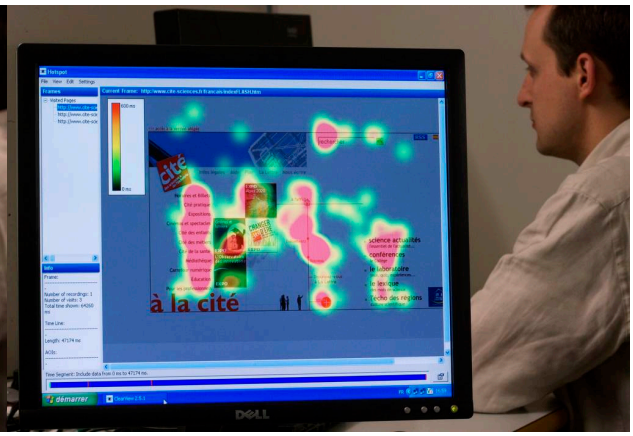


Côté observateurs

Côté participants

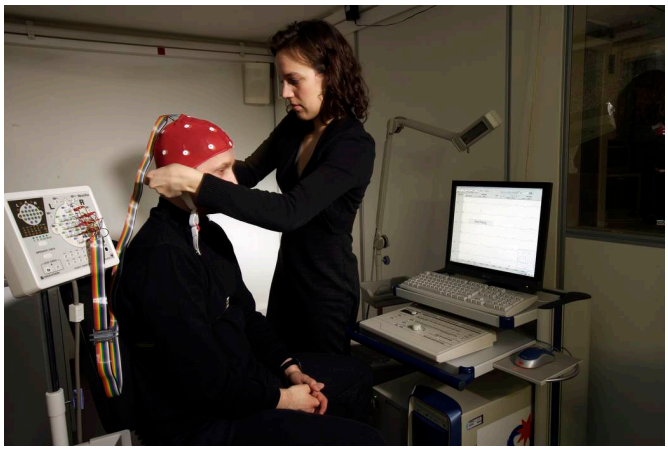
Usability Lab (LUTIN à La Villette)

Eye tracking



Heat map

Usability Lab (LUTIN à La Villette)



Mesures physiologiques (émotions)

En situation de mobilité

Styles d'interaction

système conversationnel (shell)

fenêtres (minitel!)

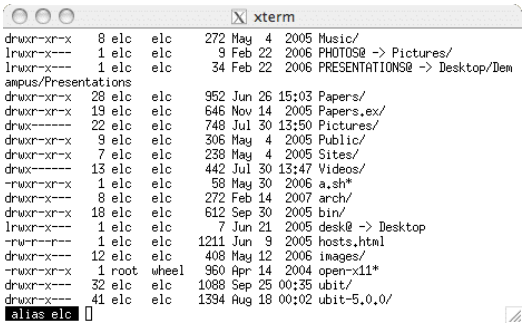
manipulation directe (WIMP)

réalité virtuelle (ou augmentéé)

Conversationnel

- **dialogue imposé par système**
- langage de commande
  exemple : Shell Unix

=> réservé aux utilisateurs experts



Styles d'interaction

Fenêtres, champs étiquetés

- **dialogue imposé par le système**
- aidé par les champs et les etiquettes
- interaction limitée
- exemple type : minitel

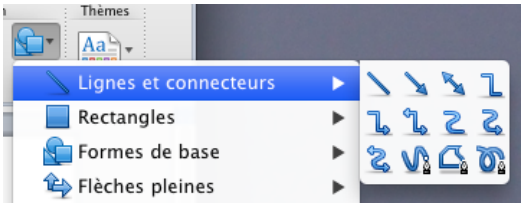
Formulaires

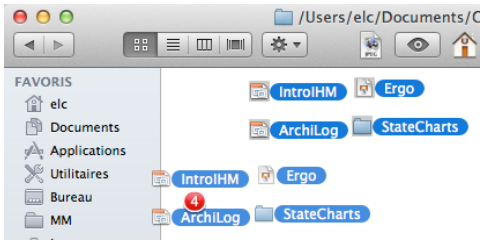
- typiquement: « Web 1.0 »
- transaction client/serveur
- contrôle a posteriori

Styles d'interaction

Modèle WIMP

- *Window, Icon, Menu, Pointer*
- **dialogue contrôlé par l'utilisateur**





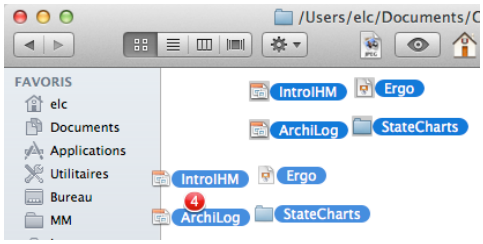
Styles d'interaction

Modèle WIMP

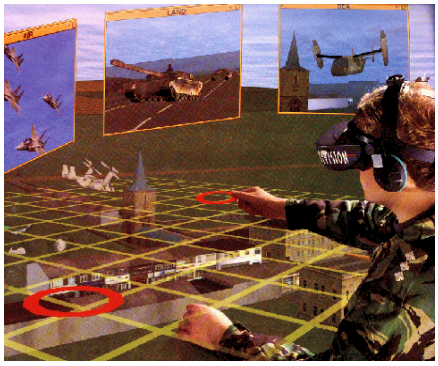
- *Window, Icon, Menu, Pointer*
- **dialogue contrôlé par l'utilisateur**

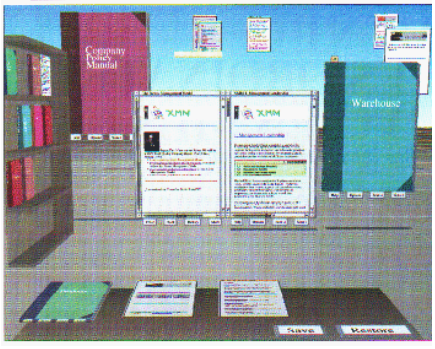
Manipulation directe

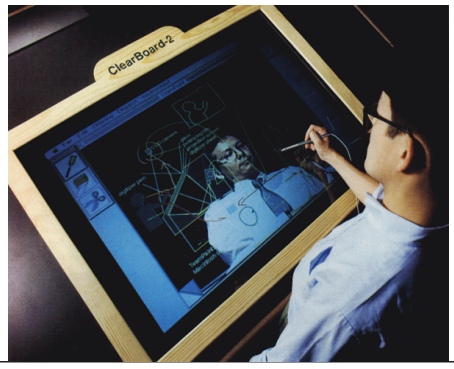
- représentation **permanente**
- actions physiques **directes**
- opérations **rapides**, **incrémentales**, **réversibles**, **feedback immédiat**
  - ⇒ apprentissage exploratoire
- **métaphores**
  - analogie, transfert d'expérience



Nouvelles techniques d'interaction



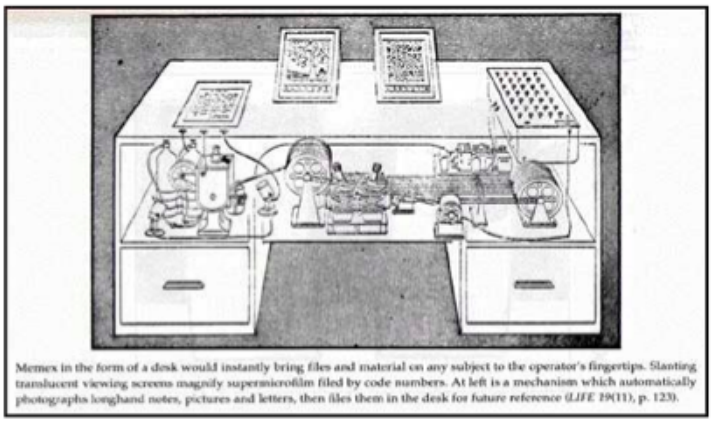




Un peu d'histoire ...

1945: Vannevar Bush (Memex)

- un instrument de mémoire externe
  - mots clés, références, indexation, liens, annotation...
- système « hypertexte » basé sur microfilms !



Memex in the form of a desk would instantly bring files and material on any subject to the operator's fingertips. Slanting translucent viewing screens magnify supermicrofilm filed by code numbers. At left is a mechanism which automatically photographs longhand notes, pictures and letters, then files them in the desk for future reference (*LIFE* 19(11), p. 123).

Un peu d'histoire ...

1963 : Ivan Sutherland (Sketchpad, MIT)

- logiciel de dessin avec icônes
- manipulation directe
- programmation OO
- PhD thesis ... visionnaire !

Ecran « haute résolution »

- oscilloscope

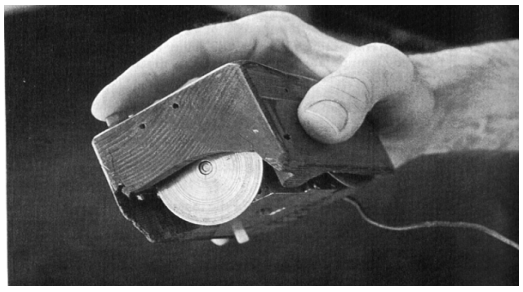
Manipulation:

- stylo optique

Un peu d'histoire ...

1962/64/... : Douglas Engelbart

- **souris**
- écran « haute résolution »
- traitement de texte,
- hypertexte et hypermedia
- messagerie électronique
- visio-conférence
- groupware ...



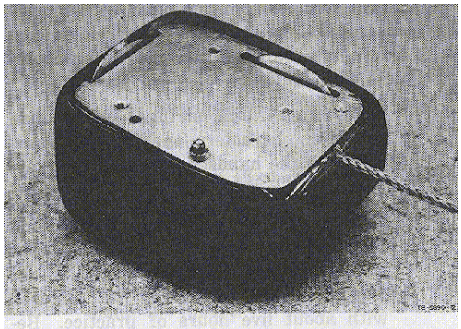
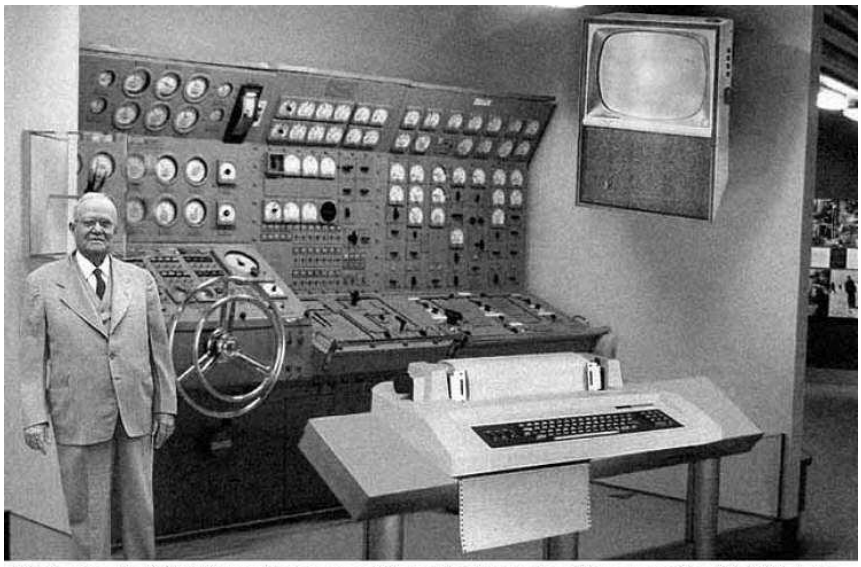


FIGURE 2—Underside of mouse

La vision du « home computer » … en 2004 !



Scientists from the RAND Corporation have created this model to illustrate how a "home computer" could look like in the year 2004. However the needed technology will not be economically feasible for the average home. Also the scientists readily admit that the computer will require not yet invented technology to actually work, but 50 years from now scientific progress is expected to solve these problems. With teletype interface and the Fortran language, the computer will be easy to use.

Un peu d'histoire ...

Xerox PARC (Palo Alto, CA)

- centre de recherche « historique », fondé en 1970
- photocopie, bureautique...

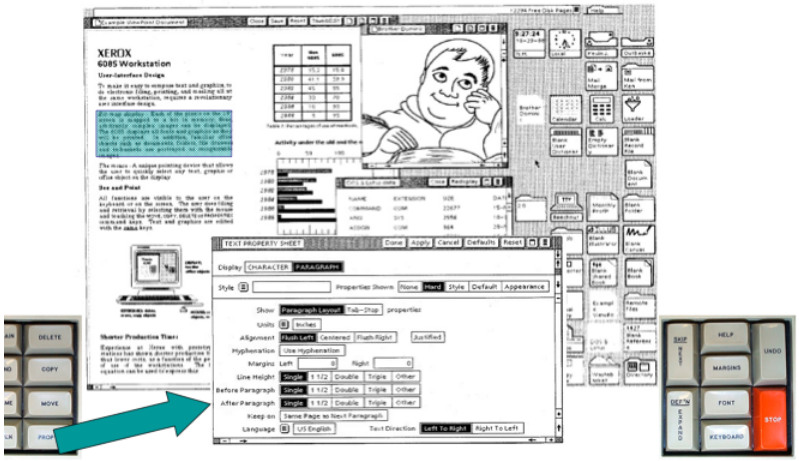
Alain Kay

- un des fondateurs du PARC
- concept de **Personal Computer** (en 1969)
- une vision radicalement différente de ce qui existait :

Un peu d'histoire ...

1981: Xerox Star

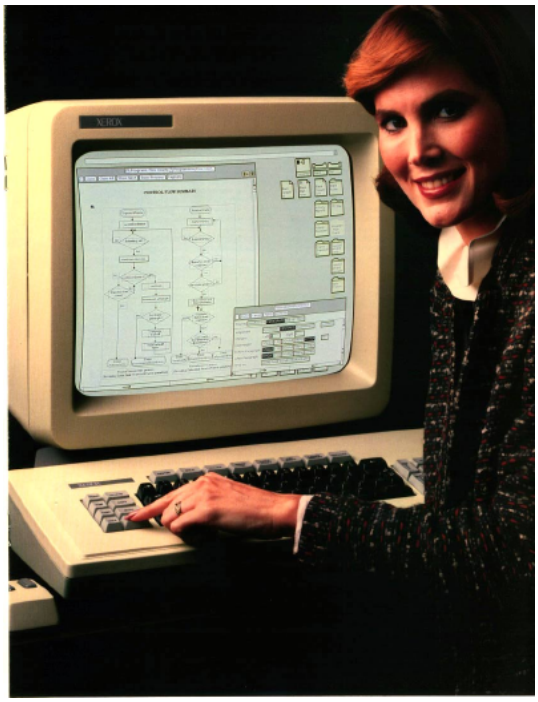
- 1er modèle commercial de station de travail
- environnement graphique évolué, Desktop, WYSIWYG



Un peu d'histoire ...

1981: Xerox Star

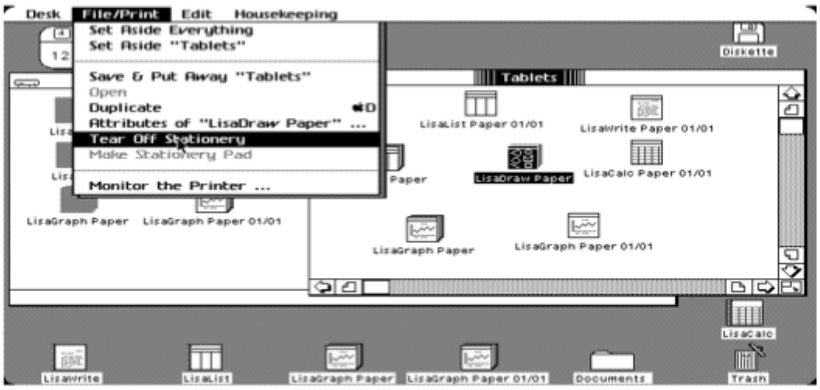
- orienté professionnels
- trop cher ($15 000 !)
- échec commercial



Un peu d'histoire ...

1983: Apple Lisa

- plus ou moins inspiré du **Xerox Star**
- plutôt un ordinateur personnel
- trop cher, échec commercial



Un peu d'histoire ...

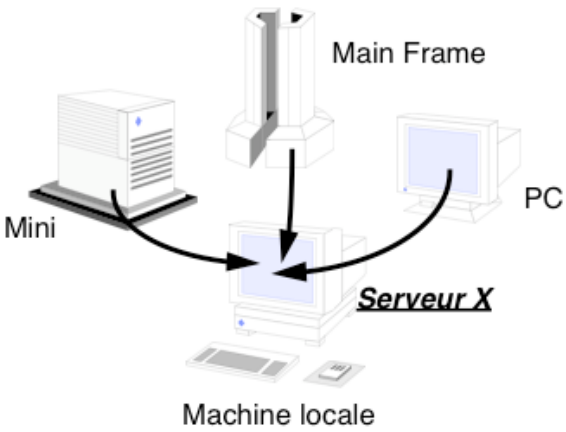
1984: Apple MacIntosh

- « rien de nouveau »
- mais bien fait et prix raisonnable ($ 2500)
- technologie "mûre", architecture ouverte
- WYSIWYG, « desktop publishing », impression laser
- guides de style (consistance)

Un peu d'histoire ...

1985: X Window System (MIT)

- pour « stations de travail »
  - Vax, HP, puis Sun
- multi-plateformes :
  - indépendant du matériel et du logiciel
- en réseau :
  - architecture client / serveur



Un peu d'histoire ...

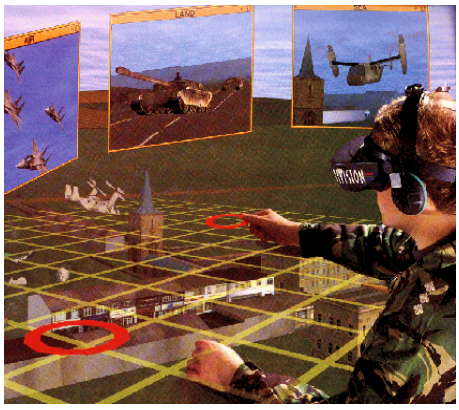
1990: World Wide Web (CERN)

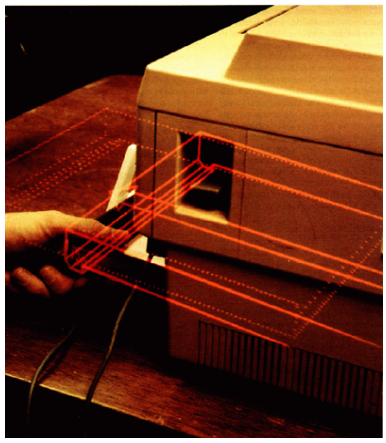
- modèle d'hypertexte en réseau
- devient hypermédia et grand public avec **Mosaic** (ancêtre de Netscape, Mozilla, Firefox)
- article refusé à la prestigieuse conférence « ACM Hypertext » !

Quoi de neuf aujourd'hui ?

Quoi de neuf ? nouveaux besoins, évolutions futures ?

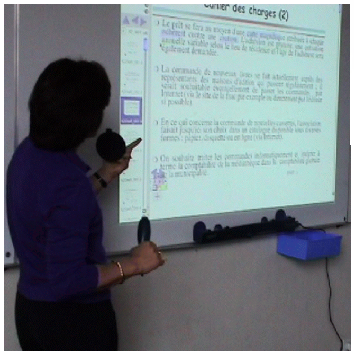
réalité virtuelle, réalité augmentée

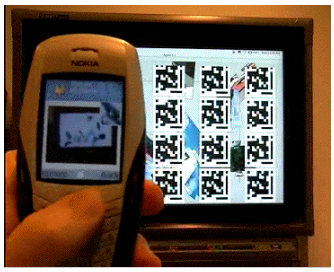
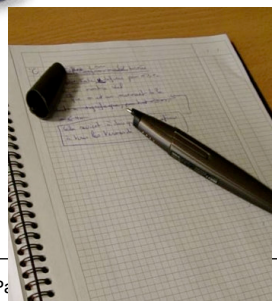




augmentation, interfaces tangibles

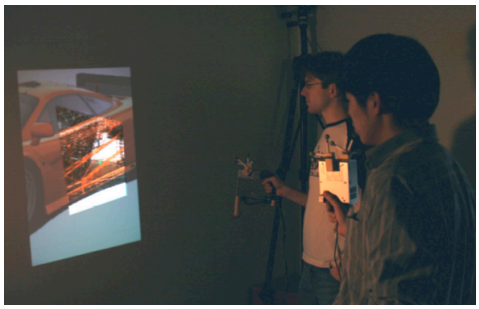




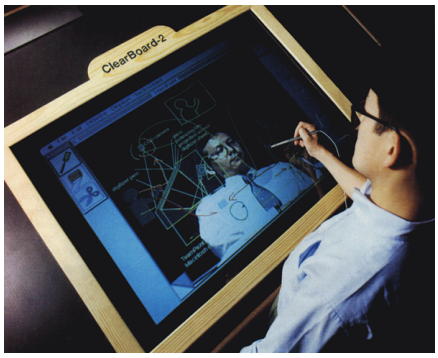



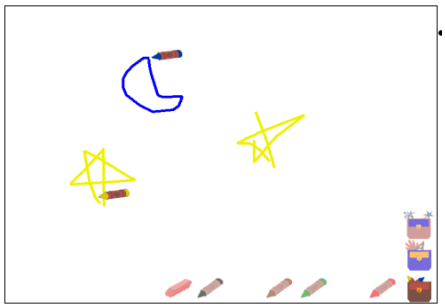
surfaces tactiles / mobiles



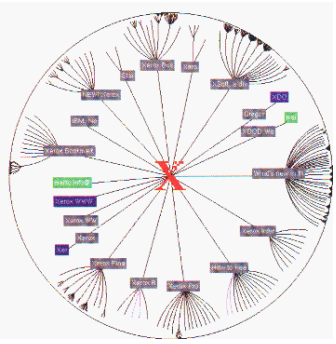



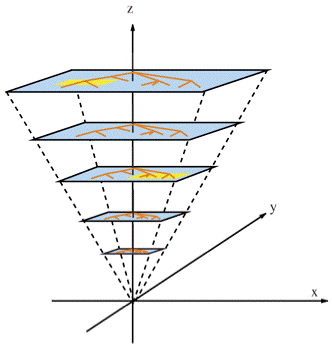
Collecticiels (groupware)

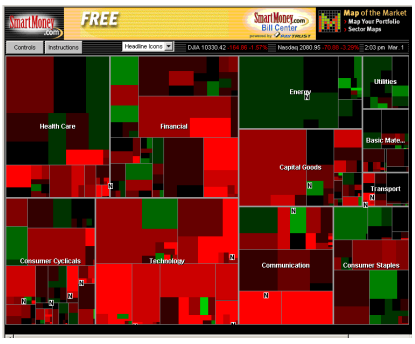




Visualisation de l'information







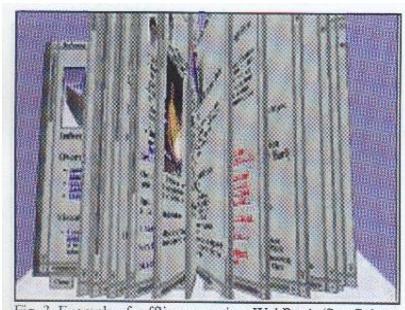


Fig. 3. Example of ruffling pages in a WebBook (See Color Plate 3)

Quelques références

Conférences

- IHM (AFIHM: **www.afihm.org**)
- ACM CHI
- UIST
- AVI ....
- INTERACT
- IEEE Visualization

....

Journaux

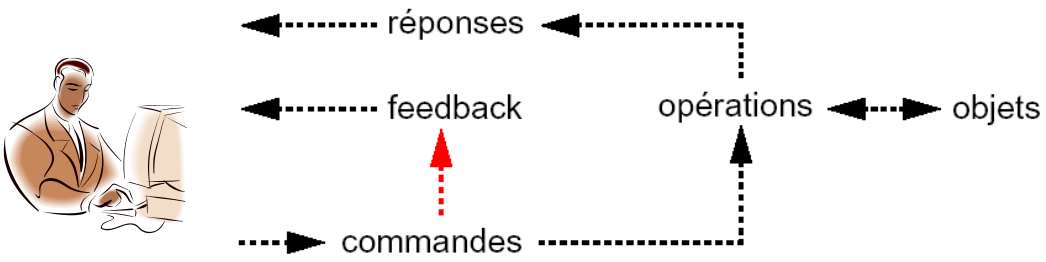
- ACM Interaction
- ACM ToCHI
- IJHCS

Conception logicielle et outils pour les IHMs

Eric Lecolinet
Télécom ParisTech
www.telecom-paristech.fr/~elc

Conception Logicielle

- **Modèle conceptuel**
  - modèle de fonctionnement du système
  - idéalement : correspond au modèle mental de l'utilisateur



src: M. Beaudoin Lafon

- **Exemple de feedback**

Langages de spécification

- **Problème**
  - spécifier de manière non ambiguë le comportement
  - en particulier pour travail en équipe !
  - génération de code éventuelle avec certains systèmes

- **Langages de description de syntaxe**
  - **UAN** : User Action Notations
  - **PPS**: Propositional Production Systems

- **Exemple PPS**
  - Color (Black, White, Red)
  - Input (**.**MouseDown, .MouseMove, .MouseUp) // noter le point
  - .MouseUp, Shift -> !DoMultipleSelect, InSelectMode

Symboles pour la réalisation des actions

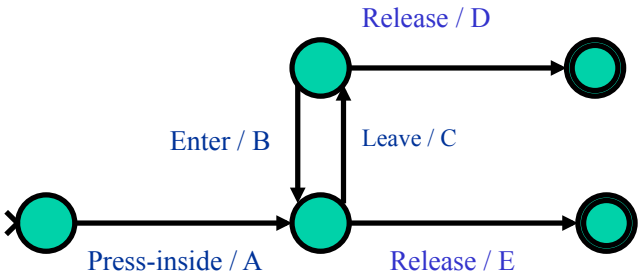
Ce que cela représente	Symbole	Sens
Mouvement du curseur	~	Déplacement du curseur
Contexte de l'objet	[X]	Contexte de l'objet X (boîte englobante)
Mouvement du curseur	~[X]	Déplacement du curseur dans le contexte de l'objet X
Mouvement du curseur	~[x,y]	Déplacement du curseur au point (x,y)
Mouvement du curseur	~[x,y]*	Déplacement du curseur au point (0,0) ou vers plusieurs points (x,y)
Mouvement du curseur	~[x',y']	Déplacement du curseur vers un point spécifique (x', y')
Mouvement du curseur	~[x,y in A]	Déplacement du curseur vers un point de l'objet A
Mouvement du curseur	~[X in Y]	Déplacement du curseur d'un point de l'objet X contenu dans Y
Mouvement du curseur	[X]~	Déplacement du curseur en dehors du contexte de l'objet X
Opération de saisie	v	Appuyer
Opération de saisie	^	Relâcher
Opération de saisie	Xv	Appuyer sur le bouton X, la touche X ou l'effecteur X
Opération de saisie	X^	Relâcher sur le bouton X, la touche X ou l'effecteur X
Opération de saisie	Xv^	Clic sur le bouton X, la touche X ou l'effecteur X
Valeur de chaine	K «abc »	Entrer la chaine abc à l'aide de l'effecteur K
Valeur de chaine	K(xyz)	Entrer la valeur dans la variable xyz à l'aide de l'effecteur K

Contraintes temporelles

Ce que cela représente	Symbole	Sens
Séquence	A B	Les tâches A et B sont réalisées en séquence, de gauche à droite ou de haut en bas
Répétition	A^*	La tâche A est réalisée 0 ou plusieurs fois
Répétition	A^+	La tâche A est réalisée au moins une fois
Répétition	An	La tâche A est réalisée exactement n fois
Option	{A}	La tâche est optionnelle : réalisée une ou 0 fois
Choix	I, OR	Choix entre deux tâches
Choix répété	$(A \mid B)^*$	Succession de choix entre la tâche A et la tâche B
Indépendance	A & B	Les tâches A et B sont réalisées en séquence sans contrainte d'ordre
Interruptabilité	A -> B	La tâche A peut interrompre la tâche B
Non interruptabilité	<A>	La tâche A ne peut pas être interrompue
Entrelacement	A ⇔ B	Entrelacement des tâches
Concurrence	A IIB	Les tâches A et B sont réalisables simultanément
Attente	A (t > n) B	La tâche B est réalisée après une attente de n unités de temps après la réalisation complète de la Tâche A

Machines à états

- **Répresentation formelle du comportement**
- **Avantages**
  - représentation graphique
  - lien avec l'implémentation, divers outils
- **Inconvénients**
  - tend à « exploser »
  - => représentations hierarchiques (statecharts...)



(à suivre)

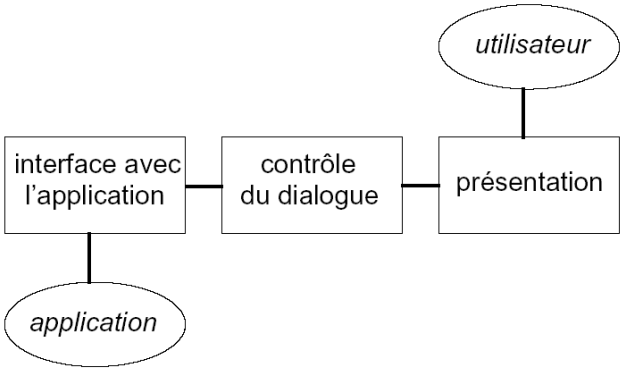
Modèles d'architecture logicielle

- **Structure générique de conception**

Modèle Langage

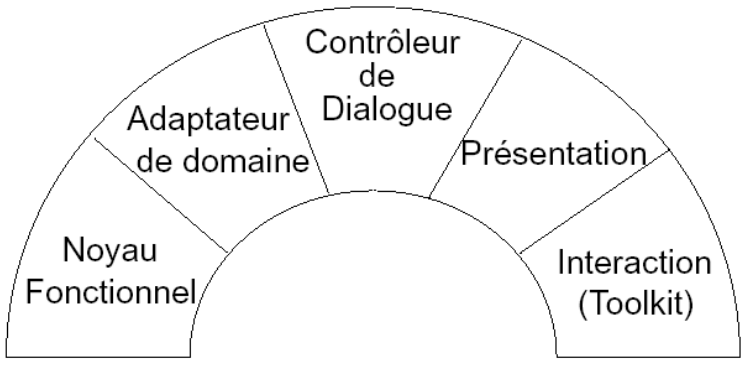
- **Vue linguistique de l'interaction**
  - analogie : IHM <=> dialogue entre individus
- **3 composantes :**
  - **Lexicale**: vocabulaire d'entités d'entrée-sortie
  - **Syntaxique:** construction des phrases
  - **Sémantique:** signification des phrases

Modèle de Seeheim



- Inspiré du modèle langage
- **Présentation**
  - primitives et objets graphiques de base (entrées et affichage)
- **Contrôle du dialogue**
  - exemples: enchaînement des écrans, sélection et modification d'objets graphiques, etc.
- **Interface avec l'application (noyau fonctionnel)**
  - relie l'interface aux concepts et objets du domaine d'application

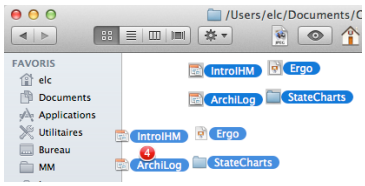
Modèle Arch

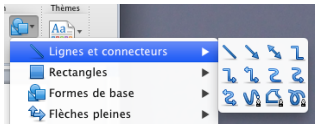


- **Extension de Seeheim**
  - **Pieds de l'arche** : composants imposés par réalité
  - **Interaction** : objets d'une boîte à outils (toolkit graphique)
  - **Présentation** : "abstraction" de la boîte à outils (-> boîte à outils virtuelle)

Contrôle du dialogue

- **Exemples de syntaxe**
  - objet **verbe**
  - (objet)* verbe
  - (objet)* verbe objet
  - verbe objet
  - verbe (objets)*
  - verbe objet *ou* (objet)* verbe

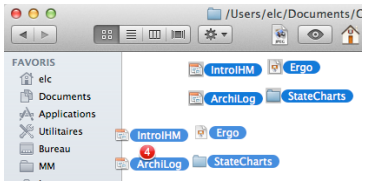


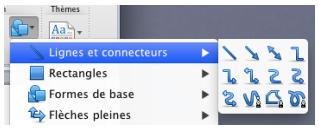


Surligner un ou plusieurs mots dans Word

Contrôle du dialogue

- **Exemples de syntaxe**
  - objet verbe
    - mettre du texte en orange
  - (objet)* verbe
    - détruire plusieurs fichiers
  - (objet)* verbe objet
    - déplacer plusieurs fichiers
  - verbe objet
    - tracer un seul trait
  - verbe (objets)*
    - tracer plusieurs traits de suite
  - verbe objet *ou* (objet)* verbe
    - le surligneur de Word





Surligner un ou plusieurs mots dans Word

Contrôle du dialogue

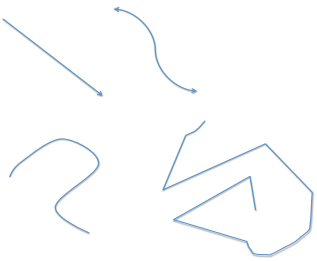
- **Modes spatiaux**
  - l'effet d'une action dépend de **l'endroit** où elle est effectuée
    - boutons, menus...
  - **avantage** : visibilité
    - découverte, mémorisation

Contrôle du dialogue

- **Modes temporels**
  - l'effet d'une action dépend du **moment** où elle est effectuée
    - palettes d'outils de dessin
    - boîte de dialogue modale
  - généralement incontournables pour créer de **nouveaux objets**
  - **inconvénient** : visibilité
    - découverte, erreurs (quel est l'état courant ?)

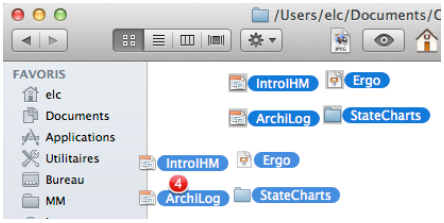
Contrôle du dialogue

- **Mode temporel**
  - actions interprétées d'une manière spécifique tant que le mode est activé
    - exple: tracer plusieurs traits
  - syntaxe : verbe (objets)*



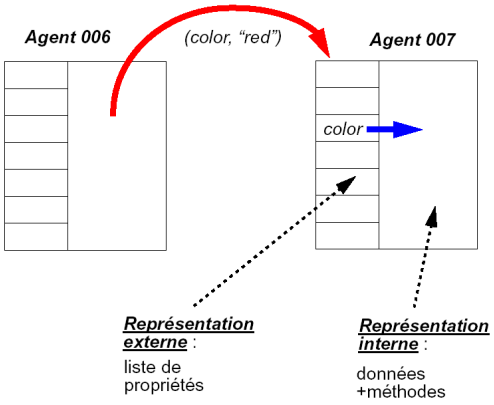
- **Cas « one shot »**
  - n'affecte que l'action suivante
    - exple: tracer **un seul** trait
  - syntaxe : verbe objet

- **Quasi-mode**
  - automatiquement désactivés à la fin de l'action
    - modifieurs, drag and drop



Modèle multi-agents

- **Vision: système interactif =**
  - ensemble d'agents spécialisés
  - qui réagissent à / produisent des événements



- **Agent = système de traitement doté**
  - d'une mémoire
  - d'un état
  - d'un "processeur" (éventuellement simulé)
  - de récepteurs-émetteurs d'événements

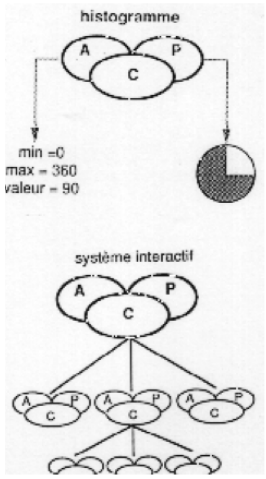
Modèles homogènes / hétérogènes

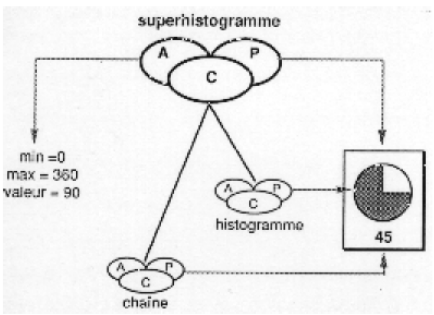
- **Modèle multi-agents = modèle général :**
  - pas d'indication sur la manière d'organiser un système interactif
- **Modèles homogènes**
  - interface = graphe de noeuds multi-facettes (multi-composantes)
  - modèle **PAC**
- **Modèles hétérogènes**
  - interface = graphe de noeuds spécialisés
  - modèle **MVC**

Modèle PAC

(Joelle Coutaz, IMAG)

- **Modèle homogène multi-agents**
- **Objets interactifs multi-facettes**
- **Structuration PAC**
  - Présentation
  - Contrôle
  - Abstraction

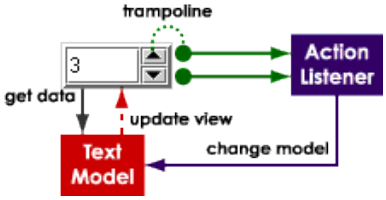


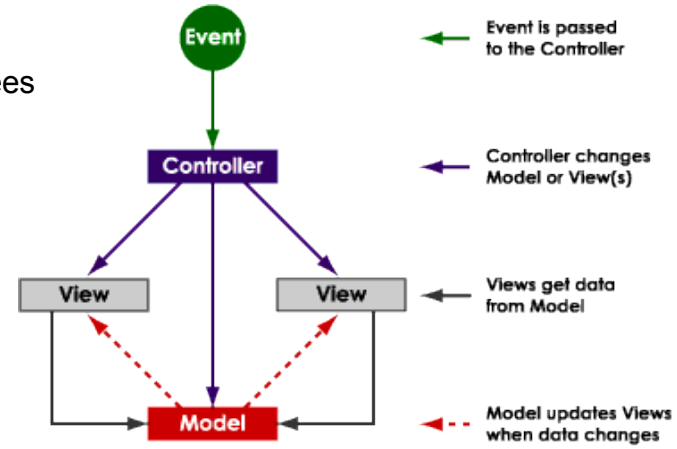


Objects composés
-- hiérarchie d'objets
-- modèle récursif

Modèle MVC

- **Trois composantes**
  - **Model** : données de l'application
  - **View** : représentation visuelle
  - **Controller** : gestion des entrées

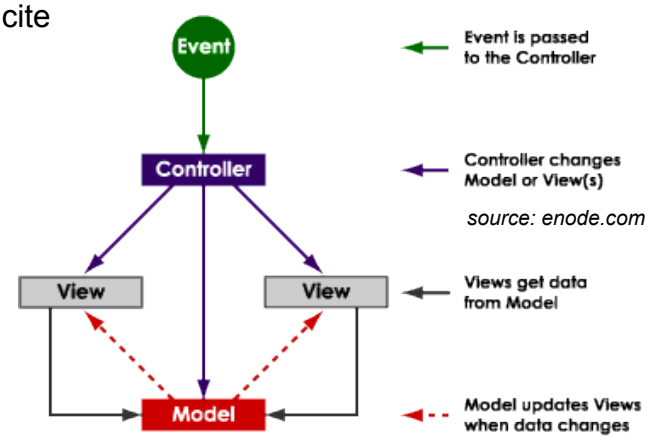




source: enode.com

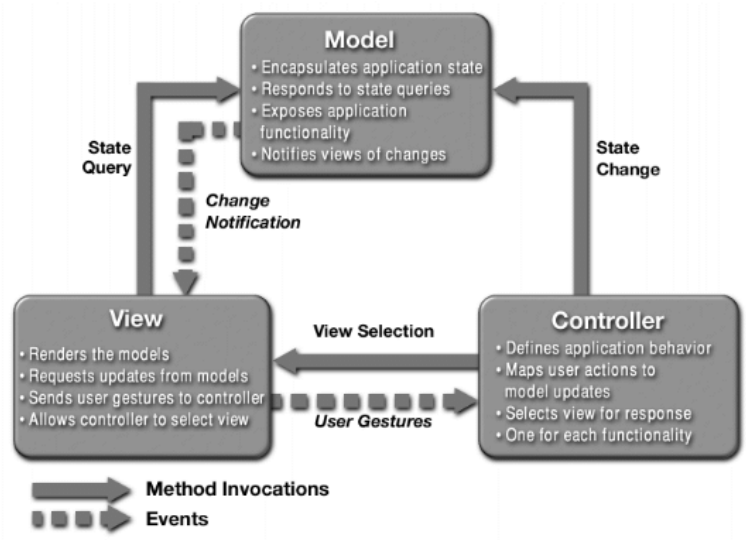
Modèle MVC

- **But de MVC**
  - mieux structurer les applications
  - représentations **multi-vues:**
    - un **modèle** peut être associé à plusieurs **vues**
    - la **synchronisation** est implicite
- **Note**
  - il existe des variantes de MVC (voir plus loin)



source: enode.com

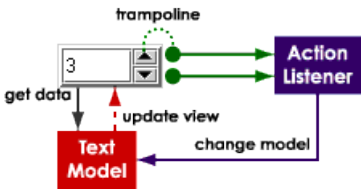
Une autre vision de MVC



source: Sun

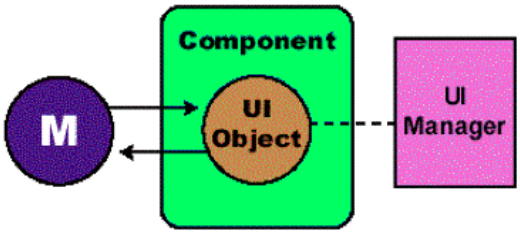
MVC et Swing

- **En pratique**
  - en pratique, **V** est fortement lié à **C** !



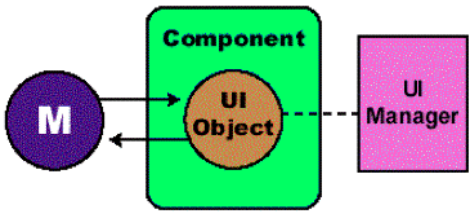
- **Swing est inspiré du modèle MVC**
  - *"Separable Model Architecture"*
  - **View** et **Controller** regroupés dans un **ComponentUI**
  - **Model** : reste séparé

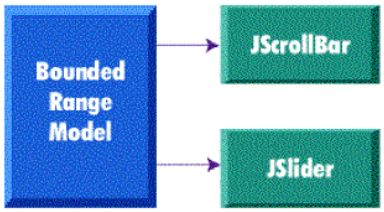
- **Un JComponent encapsule**
  - un **ComponentUI**
  - et un **Model** (si ça a un sens)



MVC et Swing

- **Modèles et multi-vues**
  - le **Model** d'un **JComponent**
  - peut être "exporté" et partagé avec celui d'un autre **JComponent**





- **Exemple**
  - **JSlider** et **JScrollbar** : même modèle : **BoundedRangeModel**
  - mise commun du modèle => **synchronisation** automatique

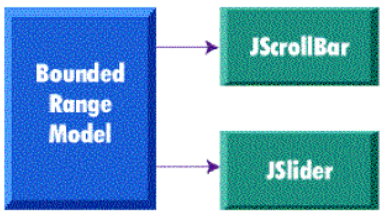
Exemple

Dans l'API de JSlider

```
public BoundedRangeModel getModel();
public void setModel(BoundedRangeModel);
```

Changer le modèle du JSlider

```
JSlider slider = new JSlider();
BoundedRangeModel model =
    new DefaultBoundedRangeModel( ) {
        public void setValue(int n) {
            System.out.println("SetValue: "+ n);
            super.setValue(n);
        }
    });
slider.setModel(model);
scrollbar.setModel(model);
```



On peut aussi ignorer l'existence des modèles

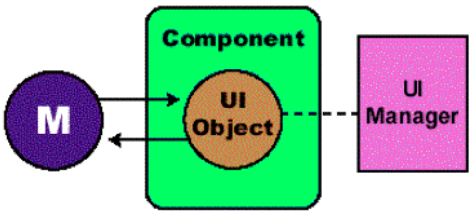
```
JSlider slider = new JSlider();
int value = slider.getValue();

// cohérent car dans l'API de JSlider on a:
public int getValue()  {return getModel().getValue(); }
```

Pluggable Look and Feel

- **"Pluggable Look and Feel"**
  - le **UIManager** peut changer dynamiquement les **ComponentUI**s

Java Metal:

```
public static void main(String[ ] args) {
    try {
      UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
    } catch (Exception e) { }
    //Create and show the GUI...
     .....
 }
```



Windows:

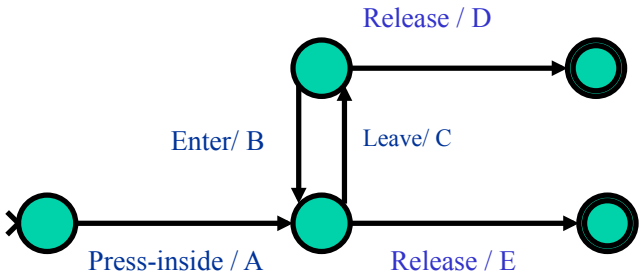
```
UIManager.setLookAndFeel(
    “com.sun.java.swing.plaf.windows.WindowsLookAndFeel”
    );
```


Outils

Outils de spécification d'interfaces

- **Grammaires**
  - langages de description de syntaxe (ex: **UAN** vu précédemment)
  - production automatique de code avec certains systèmes

- **Machines à états finis (FSM)**
  - méthode simple et efficace pour modéliser les comportements
  - outils pour faciliter l'écriture du code (SwingStates, etc.)
  - ex: cliquer sur un bouton
    (voir détails plus loin)

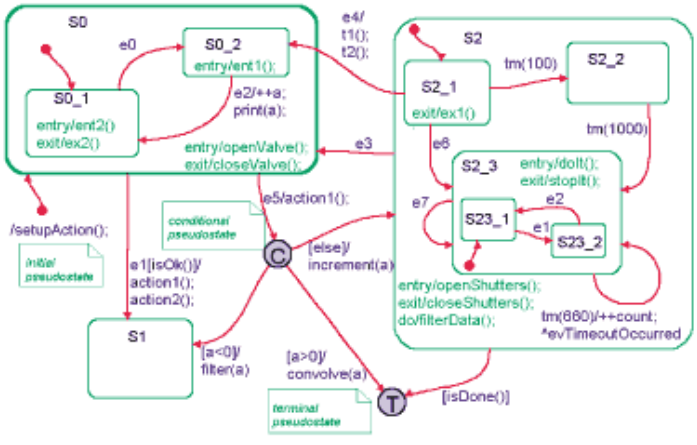


source: S. Hudson

Outils de spécification d'interfaces

- **Critique des FSM**
  - représentation "plate" => rapidement gigantesque et peu compréhensible!

- **Statecharts**
  - représentation hiérarchique :
    - fournir plusieurs niveaux d'abstraction
    - pouvoir cacher les détails



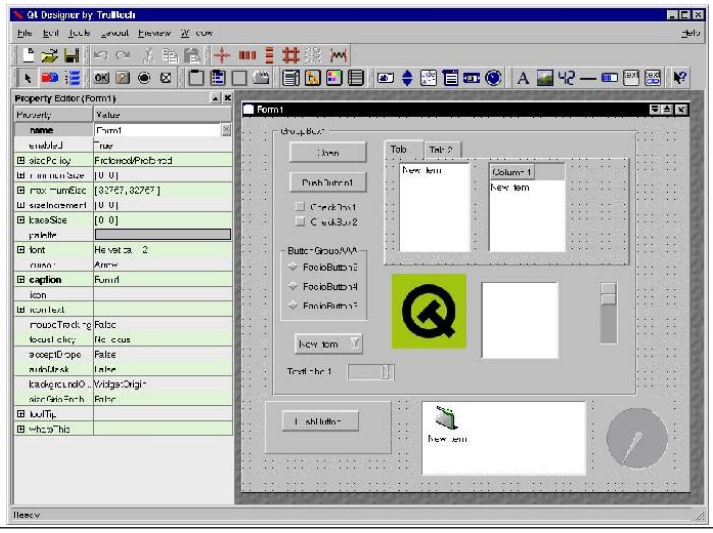
source: Shneiderman & Plaisant

- **Remarques**
  - supportés par **UML** et outils connexes
  - autre technique : réseaux de Pétri (pour vérification automatique)

Outils de génération d'interfaces

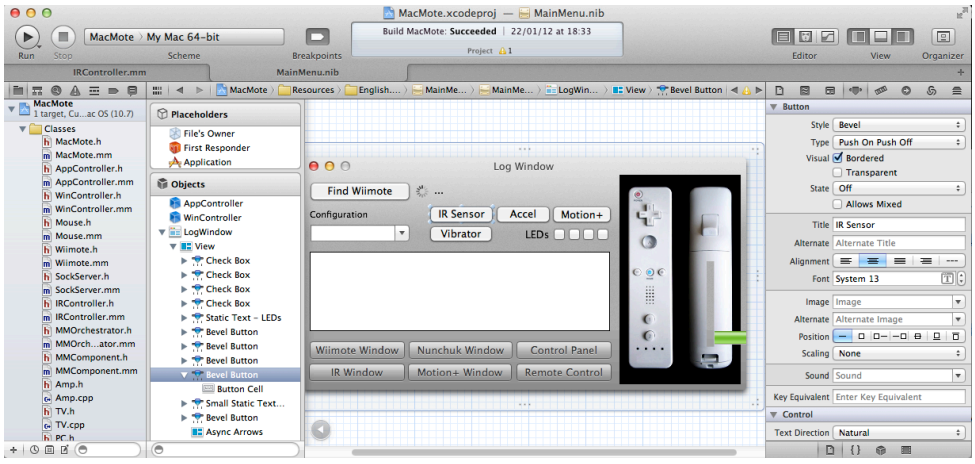
- **Outils interactifs de construction d'interfaces**
  - programmation visuelle
  - Qt Creator, Eclipse, NetBeans, XCode ...

- **Avantages**
  - prototypage
  - utilisables par des non informaticiens



Outils de génération d'interfaces

- **Inconvénients**
  - gèrent essentiellement la présentation statique
  - peu de support pour la manipulation directe
  - génération de code dans un seul sens



XCode (MacOSX)

Configurabilité

- **« Localisation »**
  - langue, nombres, dates, monnaie...
  - mots plus ou moins longs selon les langues :
    - langues orientales :
      texte ~1/3 plus **petit** que l'anglais
    - français, allemand :
      texte ~1/3 plus **grand** que l'anglais

```
dhcpwifi-22-132:~> locale
LANG="fr_FR.UTF-8"
LC_COLLATE="fr_FR.UTF-8"
LC_CTYPE="fr_FR.UTF-8"
LC_MESSAGES="fr_FR.UTF-8"
LC_MONETARY="fr_FR.UTF-8"
LC_NUMERIC="fr_FR.UTF-8"
LC_TIME="fr_FR.UTF-8"
LC_ALL=
```

=> Fort impact sur la disposition spatiale

Configurabilité

- **« Localisation »**
  - langue, nombres, dates, monnaie...
  - mots plus ou moins longs selon les langues :
    - langues orientales :
      texte ~1/3 plus **petit** que l'anglais
    - français, allemand :
      texte ~1/3 plus **grand** que l'anglais

```
dhcpwifi-22-132:~> locale
LANG="fr_FR.UTF-8"
LC_COLLATE="fr_FR.UTF-8"
LC_CTYPE="fr_FR.UTF-8"
LC_MESSAGES="fr_FR.UTF-8"
LC_MONETARY="fr_FR.UTF-8"
LC_NUMERIC="fr_FR.UTF-8"
LC_TIME="fr_FR.UTF-8"
LC_ALL=
```

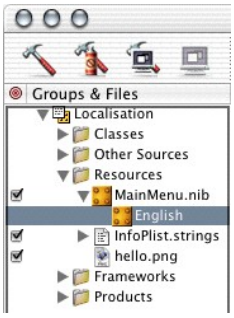
- **Accessibilité, préférences utilisateurs**
  - acuité / déficiences visuelles
  - résolution dpi : taille des pixels

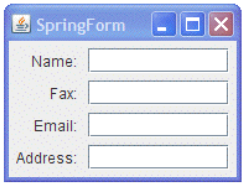
  => permettre à l'utilisateur de changer les tailles des police (entre autres)

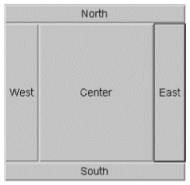
=> Fort impact sur la disposition spatiale

Configurabilité

- **Disposition spatiale : deux approches**
- **1) Ajustement « à la main » des positions et tailles des widgets**
  - une version de l'interface ajustée pour chaque langue
  - gestionnaire de fichiers de ressources localisés

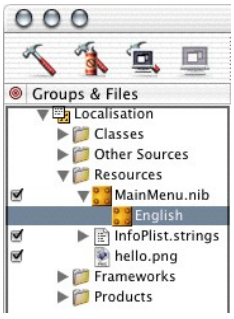


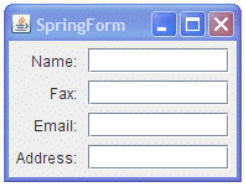


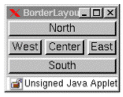


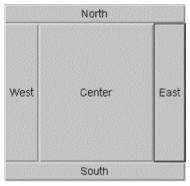
Configurabilité

- **Disposition spatiale : deux approches**
- **1) Ajustement « à la main » des positions et tailles des widgets**
  - une version de l'interface ajustée pour chaque langue
  - gestionnaire de fichiers de ressources localisés
- **2) Gestionnaires de contraintes spatiales (layout managers)**
  - calcul automatique des positions et des tailles
  - plus simple mais moins joli ?



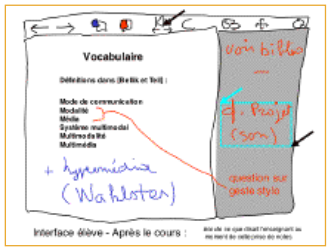




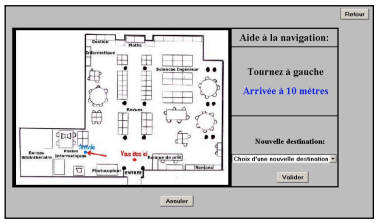


Autres outils "interactifs"

- **Mockups d'interfaces**
  - prototypes non opérationnels
    - interfaces papier, post-its
    - outils de présentation ou d'animation
    - *ex: Powerpoint, Flash...*



source: C.Faure



source: projets IAD

source: P.Baudisch

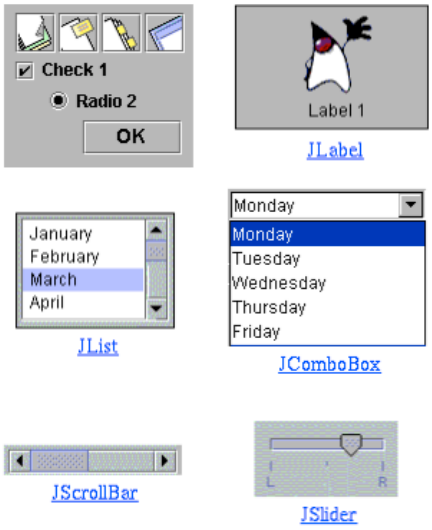
Outils de programmation

- **Langages spécialisés**
  - prototypes, petits outils, pour le Web, etc...
  - HTML/JavaScript
  - XML/UIMS et autres langages déclaratifs
  - Flash, Silverlight, Tcl/Tk, etc.

- **Boîtes à outils (toolkits) graphiques**
  - grande flexibilité
  - mais difficulté d'apprentissage !
    - réservés aux professionnels

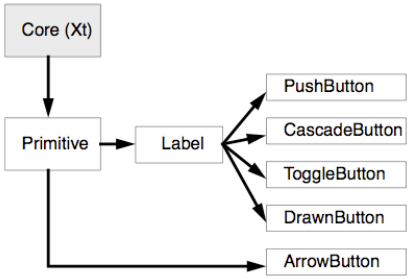
Toolkits à widgets

- **Widget = objet graphique**
  - ou « **Control** » ou « **Component** »
- **Exemples de toolkits à widgets**
  - **Smalltalk** : MVC (historique)
  - **Objective C** : Next, MacOSX
  - **Java** : AWT, SWT, Swing
  - **C++** : Qt, Visual C++
  - **C** : Gtk, Motif



Arbre d'héritage

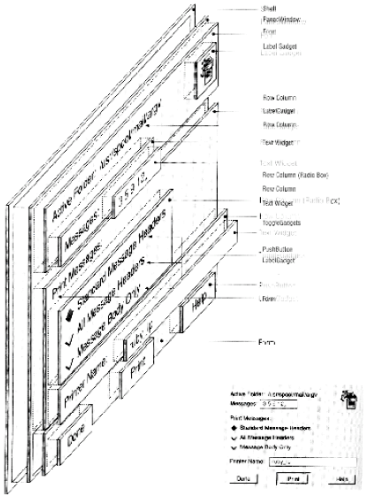
- **Hiérarchie de classes (= de types)**
  - chaque sous-classe **hérite** des variables et méthodes de sa **superclasse**
  - du plus général au plus particulier
  - héritage **simple** (*Java*) ou **multiple** (*C++*)

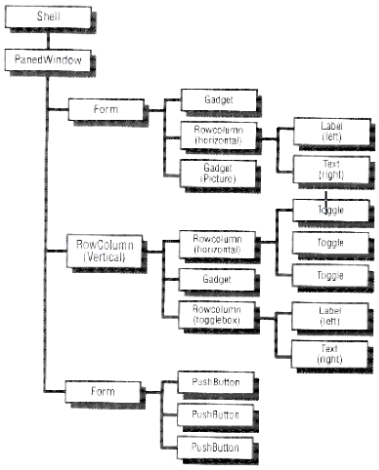


Exemple: Toolkit Motif

Arbre d'instanciation

- **Chaque objet « contient » ses enfants**
  - **clipping** : enfants (graphiquement) **inclus** dans parents
  - **superposition** : enfants **au dessus** des parents





Avantages/inconvénients des Toolkits

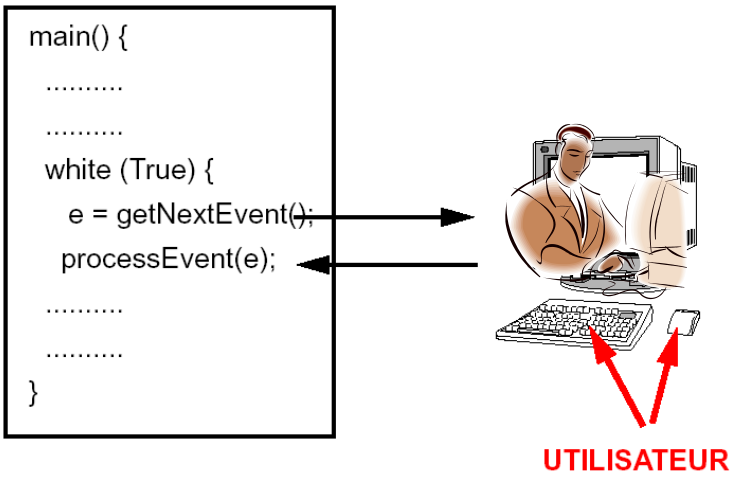
- **Avantages**
  - Portabilité informatique (matérielle, logicielle)
  - Souplesse d'utilisation
  - Extensibilité, flexibilité
  - Intégration de critères ergonomiques

- **Inconvénients**
  - apprentissage
  - effort de développement important
  - pas d'indication sur l'architecture logicielle
  - seul développeur = programmeur

Programmation événementielle

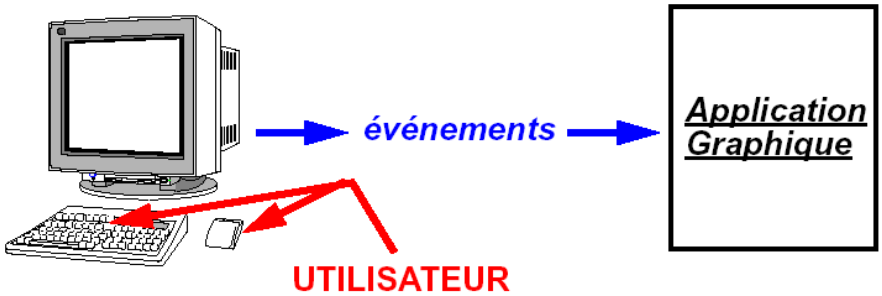
- **Interfaces graphiques : l'utilisateur a le contrôle**

  - on peut à tout moment sélectionner, cliquer, changer de fenêtre ...
  - application "esclave" de l'utilisateur
  - dialogue **multi-fils**, programmation **non modale**



Programmation événementielle

- **Conséquences**
  - application toujours prête à réagir
  - programmation **événementielle**



- **Evénement (ou « message »)**
  - envoyé à l'application ciblée
  - à chaque action élémentaire de l'utilisateur

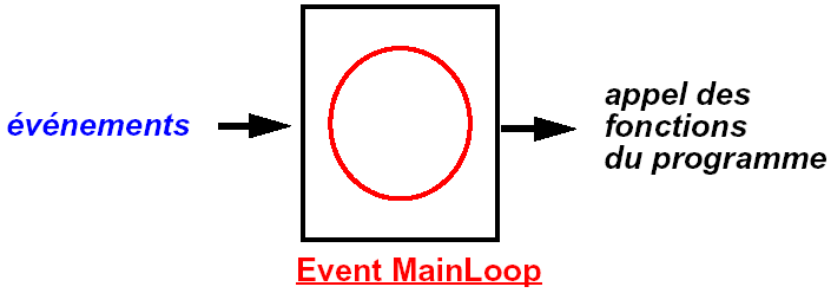
Exemples d'événements

Quelques événements X11 :

- ButtonPress, ButtonRelease
  - appuyer / relâcher un bouton de la souris
- KeyPress, KeyRelease
  - appuyer / relâcher une touche du clavier
- MotionNotify
  - bouger la souris (avec un bouton enfoncé)
- EnterNotify, LeaveNotify
  - la souris entre dans / sort d'une fenêtre
- Expose
  - rafraîchir la fenêtre (la fenêtre redevient visible)
- ResizeRequest
  - la fenêtre a changé de taille
- MapNotify, UnmapNotify
  - la fenêtre apparaît / disparaît de l'écran

Boucle de gestion des événements

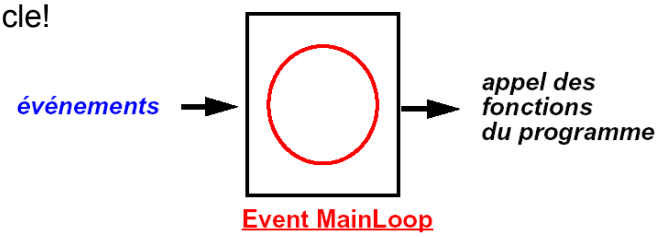
- **Boucle infinie qui**
  - récupère les événements
  - appelle les fonctions du programme



- **Deux stratégies**
  - protocole **non embarqué** (technique de base)
  - protocole **embarqué** (tous les toolkits récents)

Protocole non embarqué

- **Principe**
  - le programmeur écrit cette boucle!
  - (au moins en partie...)

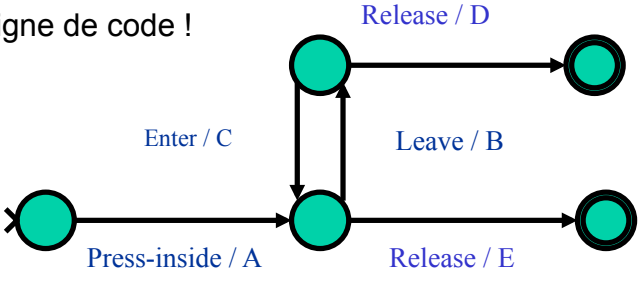



```
while (True) {
    event = GetNextEvent()
    switch(event->type) {
    case EVENT_E1 sur window_W1:  // pseudo-code
         foo_E1_W1();
         break;
    case EVENT_E2 sur window-W2:
         foo_E2_W2();
         break;
    ......
```


Protocole non embarqué

- **En réalité c'est bien plus complexe !**
  - prendre en compte :
    - **tous** les objets graphiques
    - **tous** les événements utiles
    - et toutes leurs **combinaisons temporelles !**

- **Exemple : cliquer sur un bouton**
  - bouton, menu item, liste, etc.
  - plusieurs centaines de ligne de code !



Protocole non embarqué

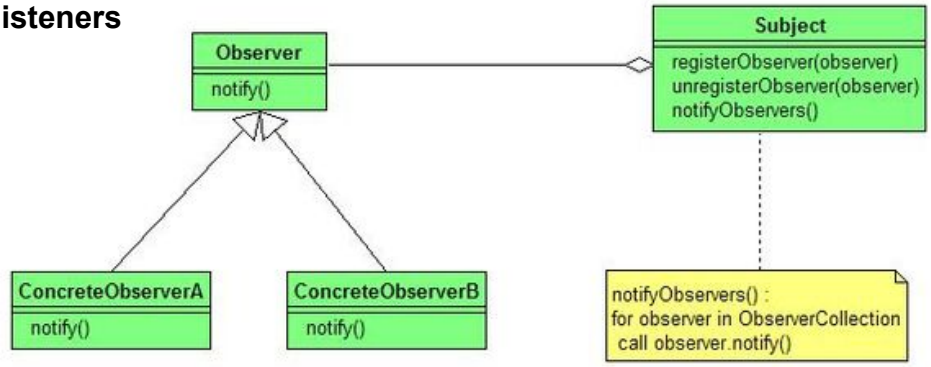
- **Conséquences**
  - complexité, risques d'erreurs

De plus:

- **Modularité, maintenance**
  - éviter **dépendance** entre objets graphiques
- **Ergonomie, utilisabilité**
  - le comportement doit être **homogène** pour toutes les applications

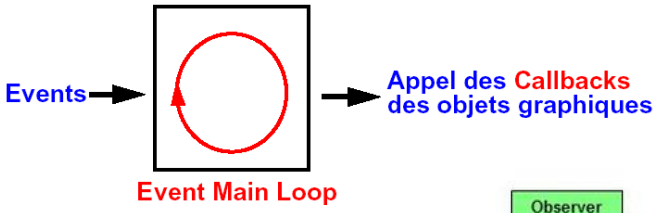
Protocole embarqué

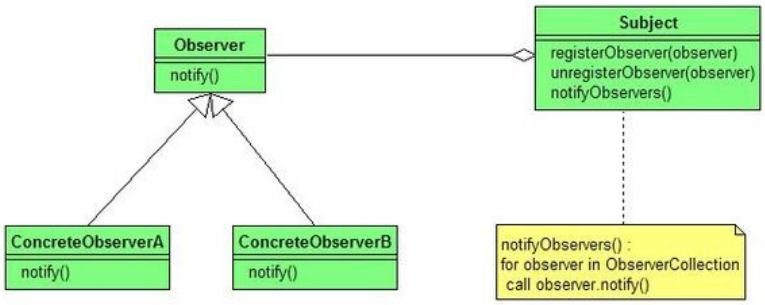
- **Principe**
  - boucle de gestion des événements **prédéfinie**
  - contrôle du dialogue **embarqué dans les objets graphiques**
- **Détection des événements**
  - via des **fonctions de callback**
  - ou des **listeners**



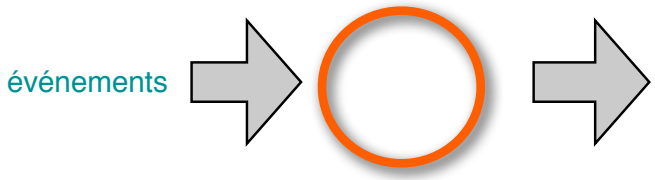
Protocole embarqué

- **Utilisation**
  - implémenter les listeners ou les callbacks
  - les associer aux widgets
  - appelés automatiquement quand la condition se produit





Modèle damaged / repaint




```
paintEvent(QPaintEvent* e) {
    QWidget::paintEvent(e);
    QPainter.painter(this);
    ......
}
```

```
mousePressEvent(QMouseEvent* e)
    ......
    ......
    update();
}
```

```
mouseMoveEvent(QMouseEvent* e)
    ......
    ......
    update();
}
```

```
mouseMoveEvent(QMouseEvent* e)
    ......
    ......
    update();
}
```

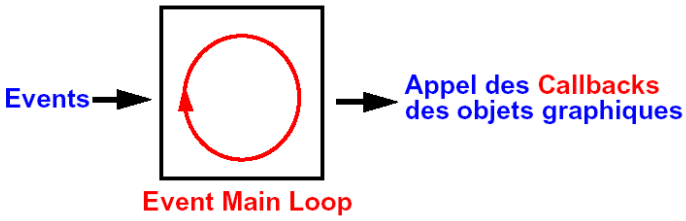
```
.......
```

```
mouseReleaseEvent(QMouseEvent* e)
    ......
    ......
    update();
}
```

paintEvent est appelée une seule fois

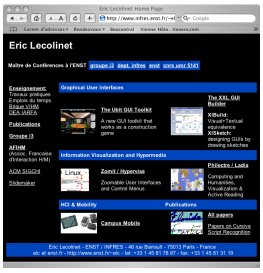
- au retour dans la boucle de gestion des événements

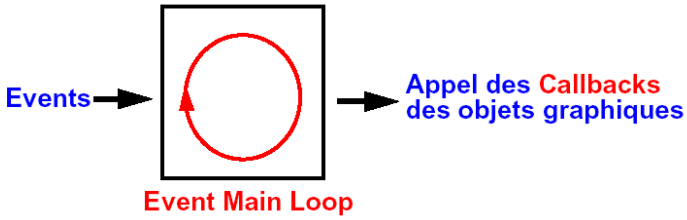
Limitations



- **Traitement séquentiel des événements**
  - la boucle de gestion des événements délègue l'activité **successivement** aux widgets graphiques
- **Conséquence ?**

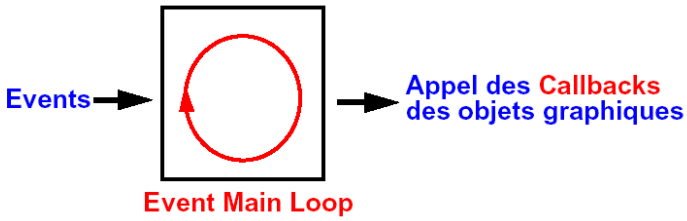
Limitations





- **Traitement séquentiel des événements**
  - si un callback bloque les événements suivants **ne sont pas traités**
  - **blocage**, plus de rafraîchissement
  - NB : les événements sont généralement mis en attente dans une pile

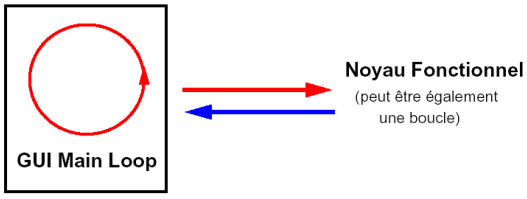
Limitations



- **Si durée d'exécution d'un callback longues ou indéterminée**
  - juste lancer un **thread** (ou équivalent) qui fait le travail
  - cas typique des **communications réseau** (sockets, etc.) !
- **Attention**
  - synchronisation toujours délicate !
  - toolkits graphiques pas toujours complètement multi-threads

Communication du NF vers la GUI

- **Notifier l'interface**
  - arrivée d'une donnée sur un pipe, une socket...
  - suivant les toolkits : thread en tâche de fond ou détection comme un événement graphique spécial



- **Cut and Paste, Drag and Drop**
  - événements spécifiques, zones tampon partagées (clipboard)
- **Cas clinique: Time Outs**
  - mise à jour à intervalle de temps régulier