Les interfaces graphiques

Principes et normes

Christian Nguyen

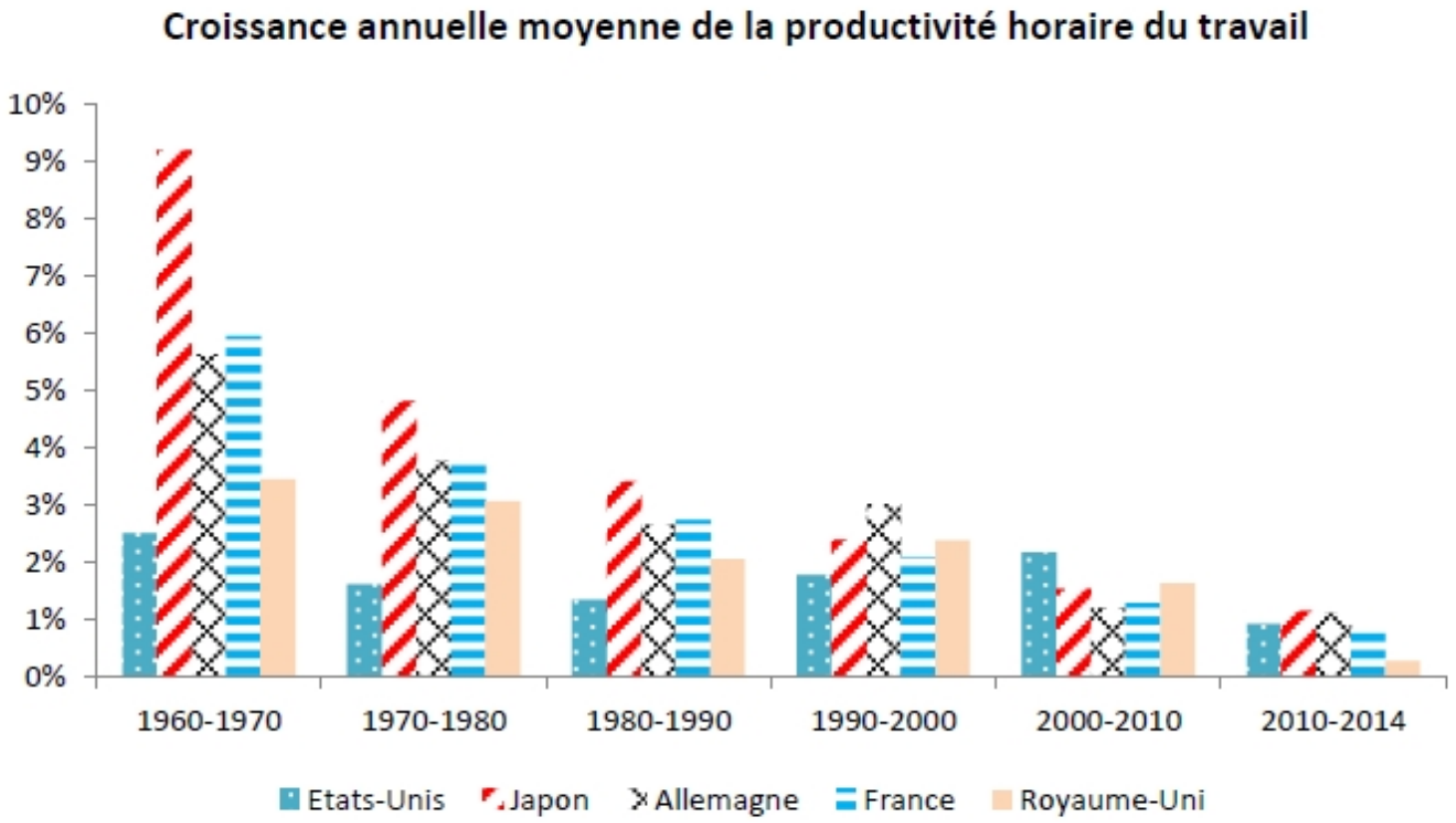
Département d'informatique
Université de Toulon et du Var

Pourquoi les IHM sont importantes ?

Panne d'ordinateur = baisse de productivité ?

Pourquoi les IHM sont importantes ?

Paradoxe de la productivité [Solow 1987] : “L’informatique se voit partout, sauf dans les statistiques de la croissance ou de la productivité.”



Source : Afep à partir de Conference Board.

Utilisabilité

Une des raisons du paradoxe : l'utilisabilité (norme ISO 9241).

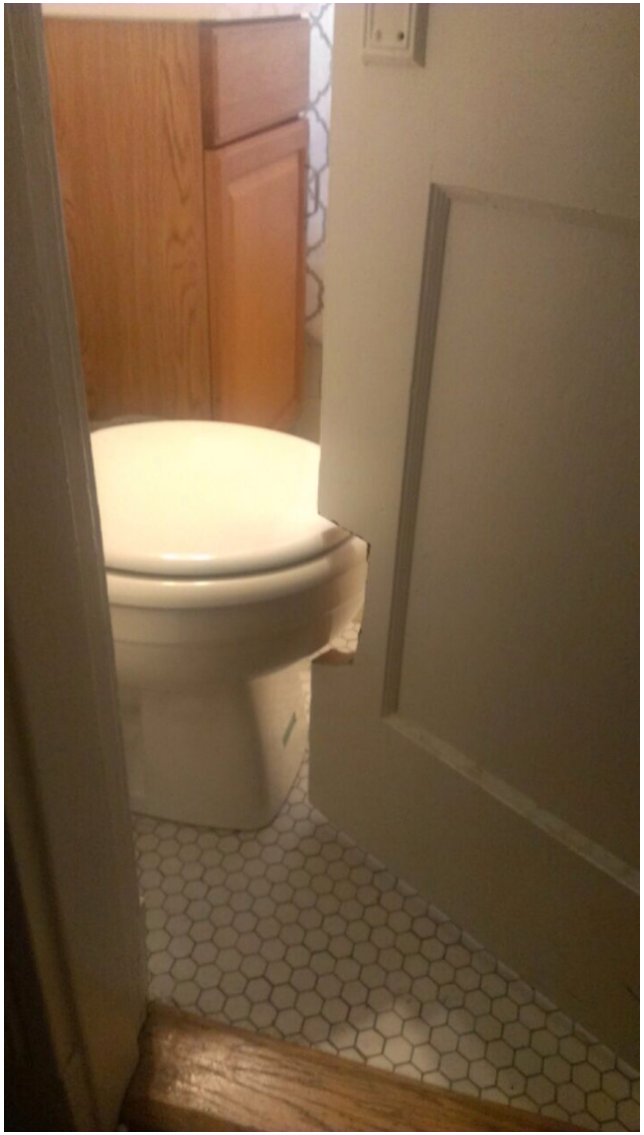
Wikipédia : degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié.

Les critères de l'utilisabilité sont :

- l'*efficacité* : le produit permet à ses utilisateurs d'atteindre le résultat prévu ;
- l'*efficience* : atteint le résultat avec un effort moindre ou requiert un temps minimal ;
- la *satisfaction* : confort et évaluation subjective de l'interaction pour l'utilisateur.

▶ solution, une conception centrée sur l'utilisateur

Utilisabilité



Utilisabilité

Utilisabilité côté logiciel interactif :

- *intuitif* : facilité avec laquelle un utilisateur peut prendre en main le logiciel et utiliser ses fonctionnalités,
- *flexible* : capacité du système à offrir des modes d'interactions multiples et à s'adapter,
- *robuste* : niveau de satisfaction dans la réalisation des tâches permises par le système (ne se limite pas à la fiabilité).

A noter que les erreurs sont la principale source de défiance vis-à-vis d'un logiciel.

Pour obtenir les fonctionnalités adéquates, une analyse complète des tâches est un préalable à la conception de l'interface.

Utilisabilité

30 concepts-clés de l'utilisabilité (30 Usability Issues To Be Aware Of - Smashing Magazine 2007) :

- Le principe des 7±2 éléments
- La règle des 2 secondes
- La règle des 3 clics
- La loi de Pareto (20/80)
- Les 8 règles d'or de la conception
- La loi de Fitts
- La pyramide inversée
- , Satisfaction
- Le syndrome de l'oisillon
- Banner blindness
- Les effets Cliffhanger et Zeigarnik
- Les lois de la Gestalt
- L'effet d'autoréférence
- L'eye-tracking (ou oculométrie)
- Le "pli" (fold)
- La zone fovéale
- Les annotations
- Dégradation élégante
- La granularité
- Les zones sensibles
- La lisibilité perceptive
- La navigation en "démineur"
- La navigation "mystère"
- La cohérence visuelle
- L'enrichissement progressif
- La lisibilité cognitive
- La conception centrée utilisateur
- La vigilance
- Le design intuitif (Walk-Up-And-Use)
- Les Wireframes

Les interfaces actuelles

Demandent à l'utilisateur de penser comme un ordinateur. Exemple : renommer un document depuis un éditeur de texte.

Placent l'utilisateur en position d'infériorité. Exemple : message d'erreur indiquant qu'il a *encore* fait une erreur.

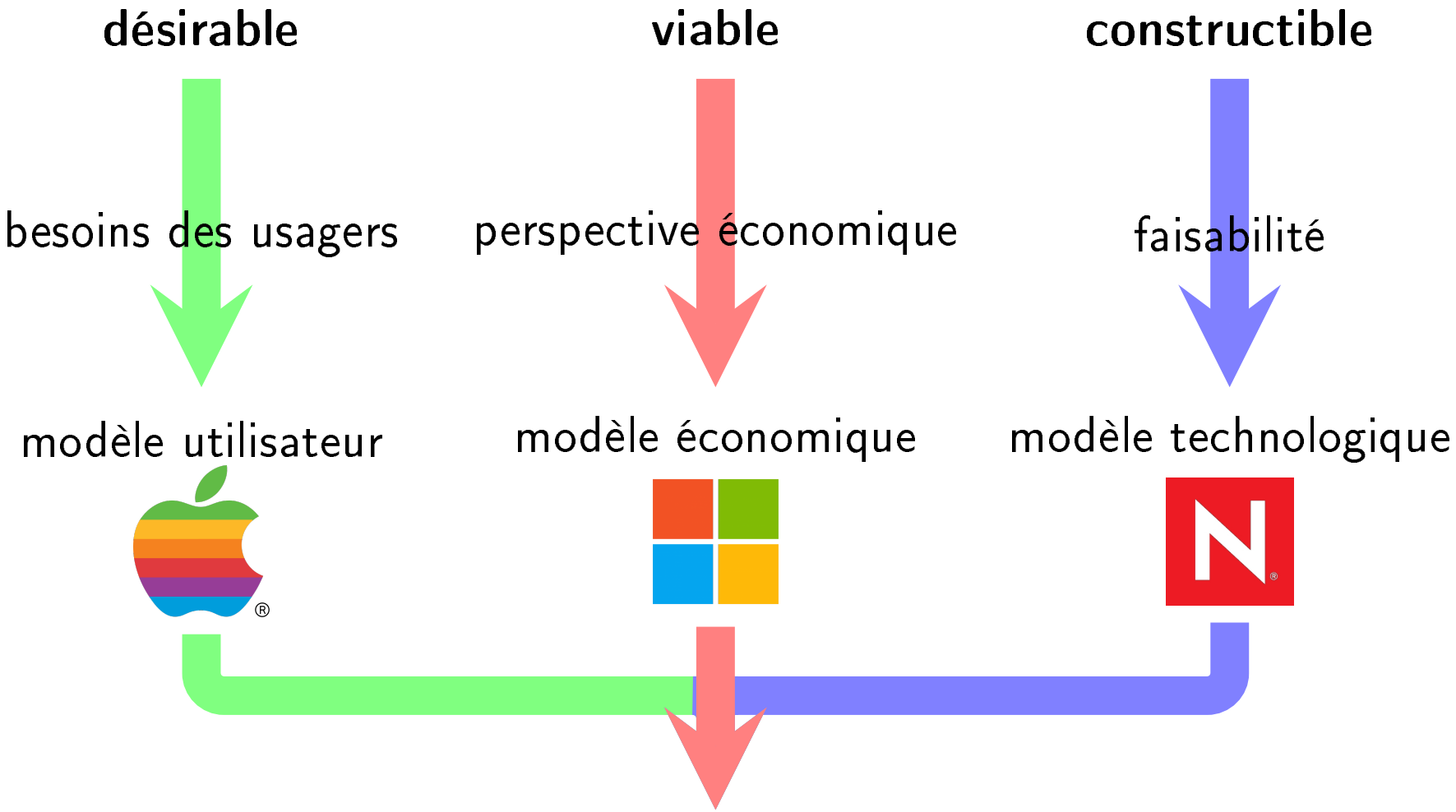
Donnent à l'utilisateur de mauvaises habitudes, liées en grande partie à une mauvaise mémorisation des faits antérieurs de la part de l'application. Exemple : sauver un document, l'imprimer, quitter.

Demandent trop d'effort à l'utilisateur pour être efficaces, en déléguant à l'utilisateur la plus grande partie du travail (qui doit gérer la logique interne de l'application).

Pourquoi un tel constat

- mauvaise priorité donnée au marketing (l'achat) face à la conception (le besoin),
- conflit d'intérêt entre conception et développement (mêmes personnes),
- ignorance quant aux véritables utilisateurs (priorités aux tendances du marché, aux contraintes techniques, ...), celui qui achète un produit n'est pas forcément celui qui l'utilise,
- absence de guide de conception (faisabilité et qualité côté techno, viabilité économique côté marketing, rien pour comprendre l'utilisateur).

La triade du développement d'un produit



Plan

Un modèle prédictif

Un individu est ramené à un système de traitement de l'information.

Trois sous-systèmes : sensoriel, moteur et cognitif.

Chaque sous-système dispose :

- d'une mémoire locale avec m sa capacité et d sa persistance,
- d'un processeur avec t son temps de cycle.

Système sensoriel

Visuel :

- $m = 17$ lettres
- $d = 200$ ms
- $t = 100$ ms (2 stimulis espacés de moins fusionnent)

Auditif :

- $m = 5$ lettres (ou équivalent)
- $d = 1500$ ms
- $t = 100$ ms

Remarque : l'intensité du stimulus agit sur la durée du cycle.

Système moteur

Un mouvement est une suite de micro-mouvements ($t = 70$ ms).

Loi de Fitts : le temps T nécessaire pour sélectionner une cible est proportionnel à la distance D à parcourir et inversement proportionnel à la taille L de la cible.

$$T = 0,1 \log_2 \frac{2D}{L}$$

Exemples :

- $D = 10$ cm et $L = 0,1$ cm donnent un temps $T = 0,8$ s,
- $D = 30$ cm et $L = 0,5$ cm donnent un temps $T = 0,7$ s,

Système cognitif

Un cycle permet la reconnaissance et entraîne l'action.

- $m = 7 \pm 2$ mnèmes [1]
- d = 10-100 ms
- t = 70 ms

Pour favoriser la mémoire à court terme (15 à 30 s) : séquence réduite de mnèmes et factorisation par motifs visuels (lettres, chiffres, mots, formes, taille, couleur, localisation) ou acoustique

1. trace laissée dans le cerveau par un événement.

Mémorisation

Conséquences : limiter le nombre de choix simultanés, établir des liens entre éléments pour faciliter le filtrage cognitif, écrire des messages concis, ne pas présenter d'informations inutiles, ...

Techniques pour favoriser la mémorisation à long terme :

- reformuler l'information,
- ajouter du sens (raconter une histoire),
- imagination visuelle (*visual thinking*),
- organiser (*chunking* : créer un mnème)
- faire des liens avec des connaissances existantes (catégories)

Pourquoi un modèle ?

- ⊕ cadre fédérateur,

  simulation sur machine,

  évaluation a priori des performances des utilisateurs.

- ⊖ performances perceptuelles et motrices, et non activités mentales de haut niveau (apprentissage, raisonnement, ...),

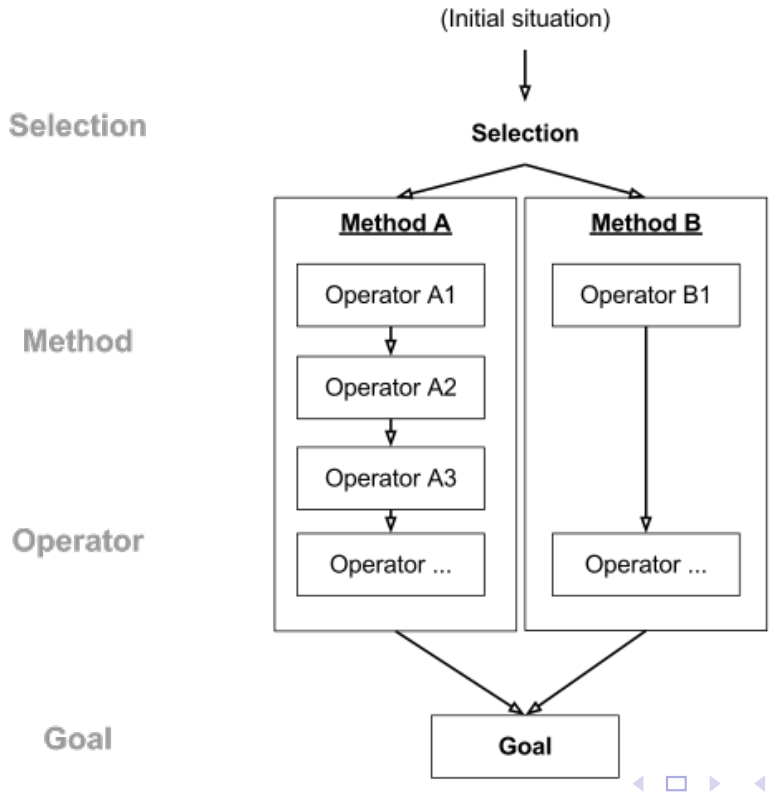
  ignore que la motricité est organisatrice de la perception (cécité corticale des chatons [Guiard 1997]),

  pas de méthode de conception.

Modèle GOMS

Acronyme de *Goals, Operators, Methods, and Selection rules*.

Modèle de description du comportement (caractère adaptatif) ; différents niveaux d'abstraction, depuis la tâche jusqu'aux actions physiques.



Modèle Keystroke

Décomposition en tâches élémentaires et génériques pour prédire le temps d'exécution.

Opérateurs :

- K (*keystroking*) : frappe, 0,2 s,
- P (*pointing*) : désignation, 0,8-1,5 s (Fitts),
- H (*horning*) : rapatriement de la main, 0,4 s,
- D (*drawing*) : dessin, $0,9n + 0,16l$ (n segments de longueur l),
- M (*mental activity*) : activité mentale, 1,35 s,
- R (*response time*) : temps de réponse du système, $\max(0, n - t)$ avec n temps de traitement et t temps exploité par l'utilisateur.

Modèle Keystroke

Principale difficulté : placer les opérateurs *M*.

Régles :

- règle 1 : insérer *M* devant chaque sous-méthode, exemple : `ls -a /usr` donne MKK MKK MKKKK
- règle 2 : supprimer *M* s'il peut être anticipé, exemple : déplacer la souris et clic donne MPMK donc MPK (car K peut être anticipé),
- règle 3 : si MKMKMKMK constitue un mot alors simplifier par MKKKK,
- etc.

Modèle Keystroke

- ⊕ analyse quantitative, permet de comparer différents choix lexicaux et syntaxiques,

  simplicité.

- ⊖ problème de la prise en compte de l'opérateur M,

  imprécision des mesures de base (moyennes ne prenant pas en compte les cas particuliers : touches spéciales, ...),

  centré sur l'aspect syntaxique et lexical de l'activité (pas de prise en compte de la complexité globale d'une tâche).

Modèle de Nielsen

Réalisation d'une tâche par un utilisateur :

1. niveau des buts : que peut-on faire dans le monde réel ?
2. niveau des tâches : que peut-on faire avec ce logiciel ?
3. niveau sémantique : que se passe-t'il si on fait telle action ?
4. niveau syntaxique : comment faire pour réaliser telle opération ?
5. niveau lexical : que signifie cet item de l'écran ?
6. niveau alphabétique : les signes,
7. niveau physique : signaux lumineux, sonores, émis.

Conception centrée utilisateur de Norman

Hiérarchisation d'une action :

,

1. définir le but
2. définir l'intention
3. spécifier une action
4. l'exécuter ▶ associations
5. percevoir l'état qui s'ensuit
6. interpréter cet état
7. évaluer le résultat ▶ retours d'information

La difficulté associée à une nouvelle expérience est directement liée aux nombres de possibilités.
▶ contraintes (physiques, logiques, sémantiques ou culturelles) pour limiter le nombre d'alternatives.

Conception centrée utilisateur de Norman

Pour savoir quoi faire :

- modèle conceptuel : fonctionnement d'un objet à partir de ses aspects visibles,
- affordance : capacité d'un objet à suggérer sa propre utilisation,
- associations (« mappings ») : relation entre contrôle et résultat,
- visibilité : rendre visibles les parties utiles d'un objet,
- retour d'information (« feedback ») : associer à chaque action un effet immédiat et identifiable.

Norman Doors

Modèle utilisateur basé archétype

Les utilisateurs étant tous différents, construction d'archétypes basés sur des études réelles et adaptés au contexte (i.e non universels).

Pas de fonctionnalités qui satisfont le plus grand nombre.

- fonctionnalités spécifiques à des besoins spécifiques.

Exemple de la voiture : décapotable - vitesse et plaisir, berline - sûr et confortable, utilitaire - fiable et robuste.

Un archétype permet d'éviter les syndrômes de(s) :

- l'homme « élastique » (polymorphe selon le concepteur),
- l'auto-référence (buts propres, modèle mental, ...),
- cas limites.

Buts vs activités

La décomposition par l'activité :

- résoud le « quoi » et non le « pourquoi » des comportements utilisateurs,
- tend à uniformiser les applications d'un domaine sans satisfaire l'utilisateur.

Un but est fonction d'une motivation qui change peu.

Une activité est basée sur la technologie, susceptible d'évolutions rapides.

Exemple : comparaison entre buts et activités d'un voyage entre hier et aujourd'hui (les buts sont toujours : rapide, confortable, sûr).

Processus mentaux

Un produit fait appel à trois niveaux de processus cognitif (Don Norman, Emotional Design, 2005) :

1. vicéral (« se sentir ») : concerne les sens, l'affect, la prise de décision rapide ▶ utilisabilité, l'utilisateur ne doit jamais se sentir stupide, en détresse, ...
2. comportemental (« faire ») : atteindre les objectifs, trouver ce que l'on recherche, résoudre les problèmes avant qu'ils ne deviennent critiques, ...
3. réflexif (« être »), mémoire des expériences passées, établissement de connexions émotionnelles fortes (Walkman, iPhone, ...).

L'homme cognitif

Activités mentales trop complexes pour élaborer un modèle général.

Notre mental crée des modèles de son environnement dont il infère ensuite des comportements.

Interface : niveau cognitif (recherche et création de modèle), niveau perceptuel (reconnaissance des formes).

Deux modèles :

- modèle conceptuel du système créé par le concepteur (à partir des modèles qu'il s'est créé de l'utilisateur au travail),
- modèle mental que se crée petit à petit l'utilisateur utilisant le système.

- point de rencontre

Interface et apprentissage

But de l'interface : faciliter l'apprentissage du logiciel i.e rapprocher le modèle mental de l'utilisateur du modèle conceptuel.

Facteurs cognitifs :

- apprentissage par l'action (essai-erreur, guidage, aide en ligne),
- apprentissage théorique (manuel rigoureux, complet, solide),
- apprentissage par analogie (métaphores, exemples).

Facteurs de personnalité :

- introvertis, extravertis (méthodes d'exploration différentes),
- inquiets (beaucoup de retour d'informations),
- impulsifs (messages courts, récupération des erreurs).

Interface et apprentissage
Typologie d'expérience

Novice : anxieux, à rassurer pour faciliter l'apprentissage et éviter les rejets.

- limiter le nombre d'actions, de concepts, feedback d'information, messages d'erreur informatifs, tutoriel, aide contextuelle, manuel d'utilisation intuitif.

Occasionnel : connaissance globale du système mais difficulté à se rappeler les différentes fonctionnalités.

- consistance de l'interface, prévention des erreurs, aide en ligne.

Expert : excellente connaissance du domaine de la tâche, du système et de son interface, recherche avant tout l'efficacité et la rapidité.

- raccourcis clavier, commande en ligne, création de macros.

Modèles d'implantation vs modèles mentaux

Exemple : l'électricité, une inversion du potentiel électrique 50 fois/s (modèle d'implantation) vs un fluide (modèle mental).

Modèle d'implantation :

- un bouton pour chaque fonction,
- un champ pour chaque entrée,
- une page pour chaque transaction,
- une boîte de dialogue pour chaque module.

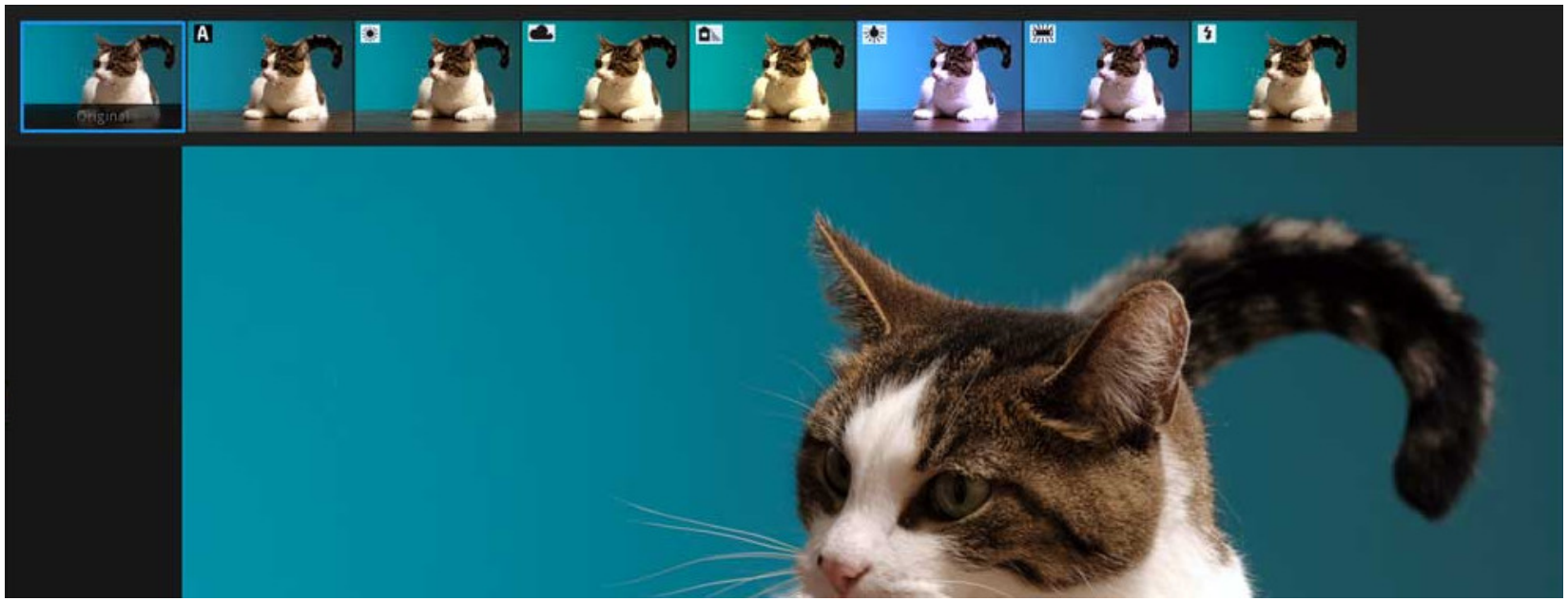
▶ pas de mécanisme cohérent pour atteindre un but.

Exemple : on peut être cinéphile et ignorer tout de la technologie associée.

Modèles d'implantation vs modèles mentaux

Le modèle de représentation doit le plus souvent être différent de la structure d'un programme.

- joue le rôle d'interface entre le modèle d'implantation et le modèle mental.



Exemple : la balance des blancs dans Photoshop Express.

Première conclusion

Hiatus entre :

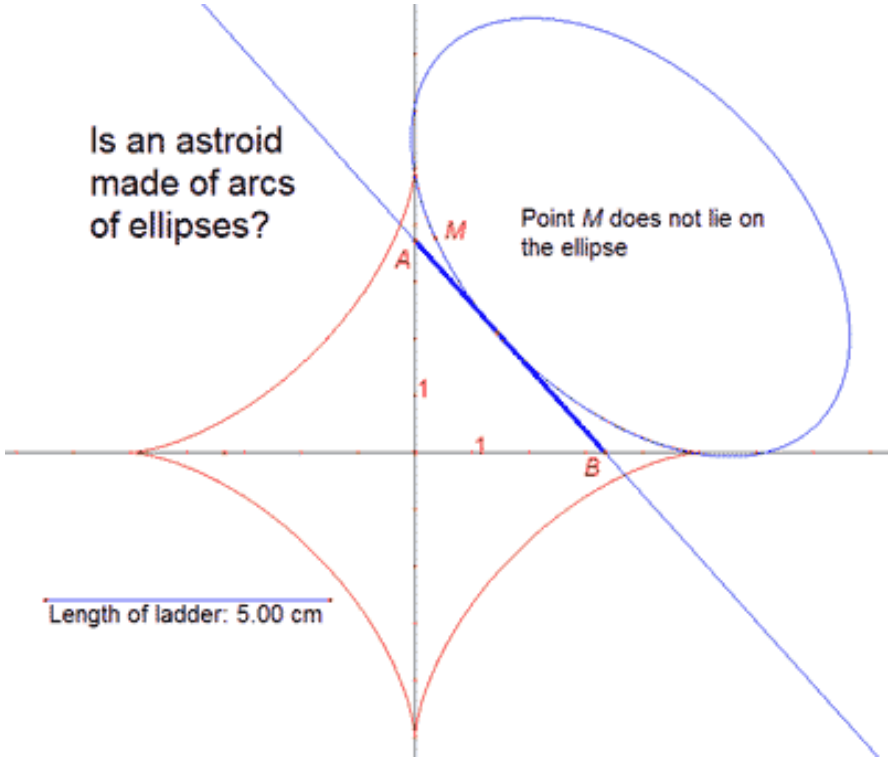
- logique de l'utilisateur (intentions et buts de haut niveau),
- limites actuelles de la logique de la machine (impose de passer par une succession de tâches de bas niveau).

Pas de modèles universels ou de théories prêtes à l'emploi mais quelques *règles* pour construire une « bonne » IHM :

- adapter la logique du fonctionnement du système informatique à la logique d'utilisation de l'utilisateur,
- créer chez l'utilisateur une logique d'utilisation par la pratique de l'interface.

Le modèle « idéal » ?

`Cabri 2` : l'application qui devine les intentions de l'utilisateur.



Exemple de démonstration sur une astroïde (hypocycloïde de cercle à quatre points de rebroussement).

Plan

Ergonomie

IEA (*International Ergonomics Association*) en 2000 :

Définition

L'ergonomie (ou *Human Factors*) est la discipline scientifique qui vise la compréhension fondamentale des interactions entre les humains et les autres composantes d'un système, et la profession qui applique principes théoriques, données et méthodes en vue d'optimiser le bien-être des personnes et la performance globale des systèmes.

Ergonomie

L'ergonomie est une science :

- apports théoriques : modèles (description, interprétation, prédiction),
- apport expérimentaux : protocoles de tests, méthodes d'analyse des données expérimentales, outils de mesure de performance, guides et principes.

L'ergonomie n'est pas intuitive :

- analyse logique de la tâche des concepteurs $\neq$ conceptualisation des utilisateurs,
- sous-estimation de la diversité des utilisateurs, de leur niveau d'expertise.

▶ objets d'études : l'utilisateur, la tâche, la communication utilisateur-machine pour effectuer la tâche.

Principes ergonomiques

La *cohérence* : un même concept doit toujours être utilisé de façon similaire dans un contexte d'utilisation identique.

La *concision* : limitation du nombre d'interventions de l'utilisateur afin d'éviter les erreurs (typographiques par exemple).

Le *retour d'informations* : toute action de l'utilisateur doit amener un retour d'information rapide et pertinent afin de lui permettre d'analyser rapidement le nouvel état de l'application.

La *structuration des activités* : l'application doit être décomposée suivant une hiérarchie de niveaux de complexité croissante.

La *flexibilité* : toute application doit pouvoir être facilement personnalisée par les utilisateurs.

La *gestion des erreurs* : l'utilisateur doit être orienté vers une méthode lui permettant de résoudre son problème.

Une interface cohérente

À but identique, séquence de commandes identiques.

Unix : incohérence lexicale pour l'opération « quitter » (•, `q`, `logout`, ...).

Si deux commandes ont des paramètres identiques, les placer dans le même ordre.

Unix : aucune logique dans les noms de commande (`cd`, `mkdir`, `grep`, `awk`, ...).

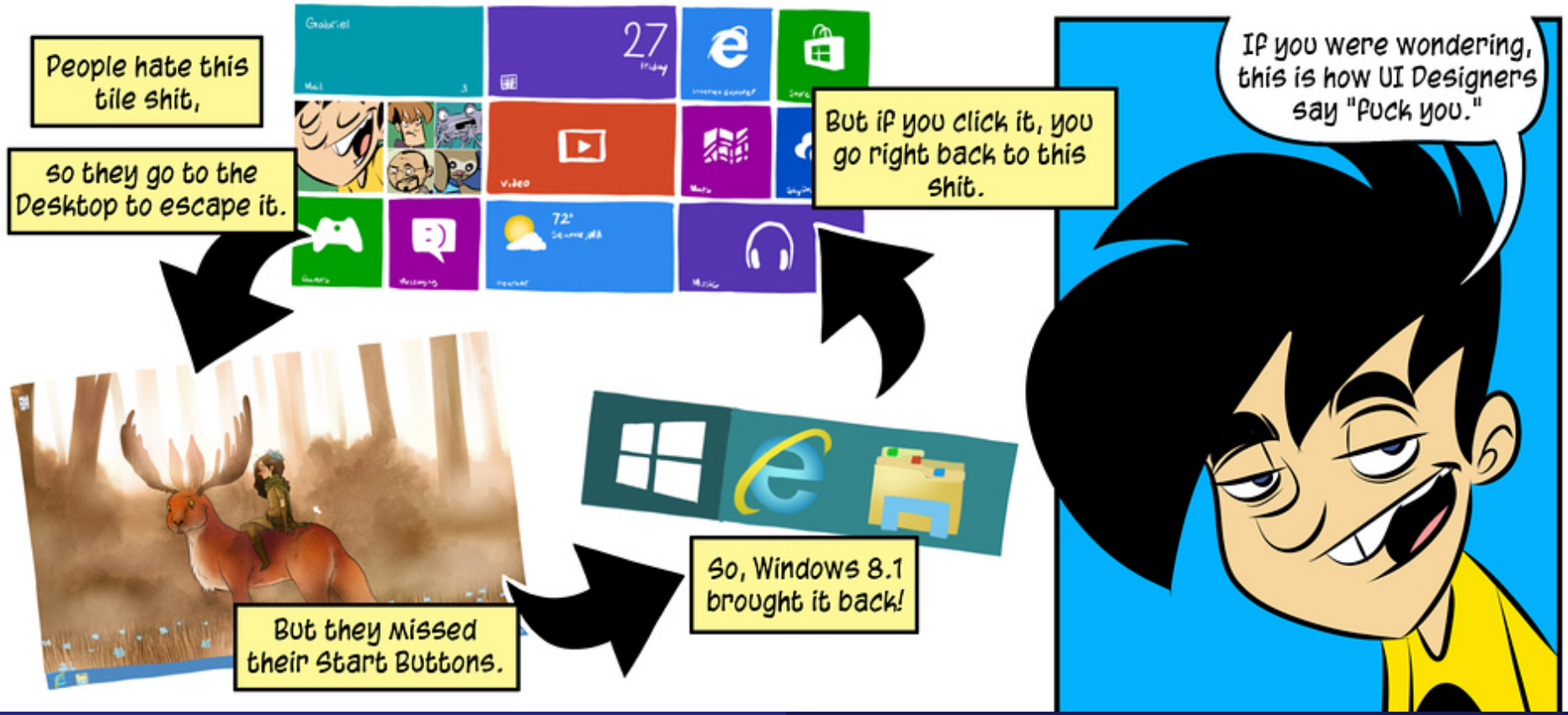
À sémantique identique, dénomination identique.

MacOS : incohérence sémantique du « drag and drop » (fichier/volume, CD/corbeille, ...).

À utilisation identique, localisation identique.

Exemple d'incohérences

Microsoft Windows 8 (Modern UI) : chaque application s'exécute en mode plein écran (multi-tâches ?), aucun moyen simple de fermer une application, Task Switcher compte le bureau comme une seule application (mais ALT-TAB non), etc.



Une interface concise

Combiner harmonieusement le bref et l'expressif (mémoire à court terme limitée !) :

- suivre des règles précises et naturelles pour les abréviations,
- donner la possibilité à l'utilisateur de créer simplement des macro-commandes,
- importation/exportation de données transparentes (par copier/coller ou autre),
- utiliser des valeurs par défaut,
- masquer les opérations impossibles dans le contexte (griser les options d'un menu, etc),
- pouvoir refaire et défaire,
- pouvoir appliquer une même opération sur différents objets.

Une interface réactive

Le retour d'information (*feedback*) doit être immédiat et informatif :

- informer pour rassurer
- informer pour réduire la charge cognitive
- informer des erreurs et indiquer un remède

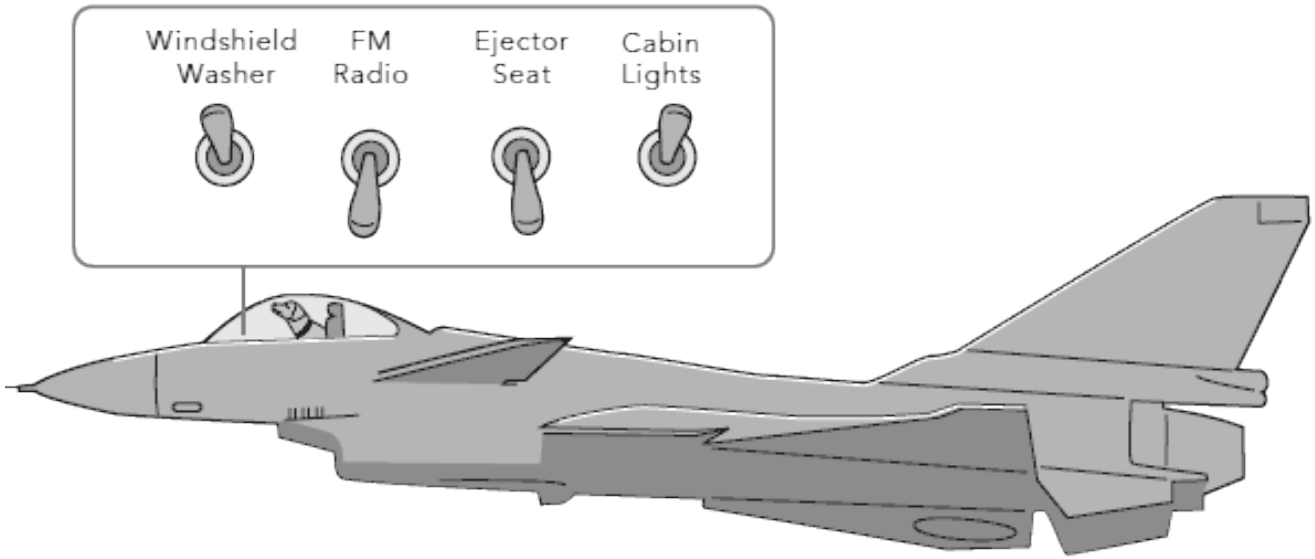
Une interface structurée

Réduire la complexité du fond (l'activité) : organiser les fonctions en niveaux de complexité croissante (*scaffolding*).

Une interface structurée

Réduire la complexité de la forme (l'image) : séparer, hiérarchiser.

Attention à l'effet « bouton du siège ejectable » ([Cooper 2007]), se souvenir de la (contraposée de) loi de Fitts.



Un exemple : KPT Texture

Une interface originale et structurée.

Une interface flexible

Une interface adaptable : modifiable par intervention explicite de l'utilisateur :

- pouvoir modifier le lexique (adapter la terminologie, reformuler les messages, redéfinir les raccourcis) facilement, sans devoir éditer de fichiers de ressources,
- pouvoir modifier les valeurs par défaut (fichiers de profil),
- représentations multiples des objets de l'interaction.

Une interface adaptative : qui se modifie automatiquement en fonction de l'activité de l'utilisateur, gestion d'un modèle de l'utilisateur par utilisation de techniques d'IA (réseaux sémantiques, réseaux bayésiens, ...).

Plan

Historique et standards

Xerox Park, 1970 : bases des *interfaces utilisateur graphiques* (IUG ou GUI en anglais) fondées sur le paradigme WIMP.

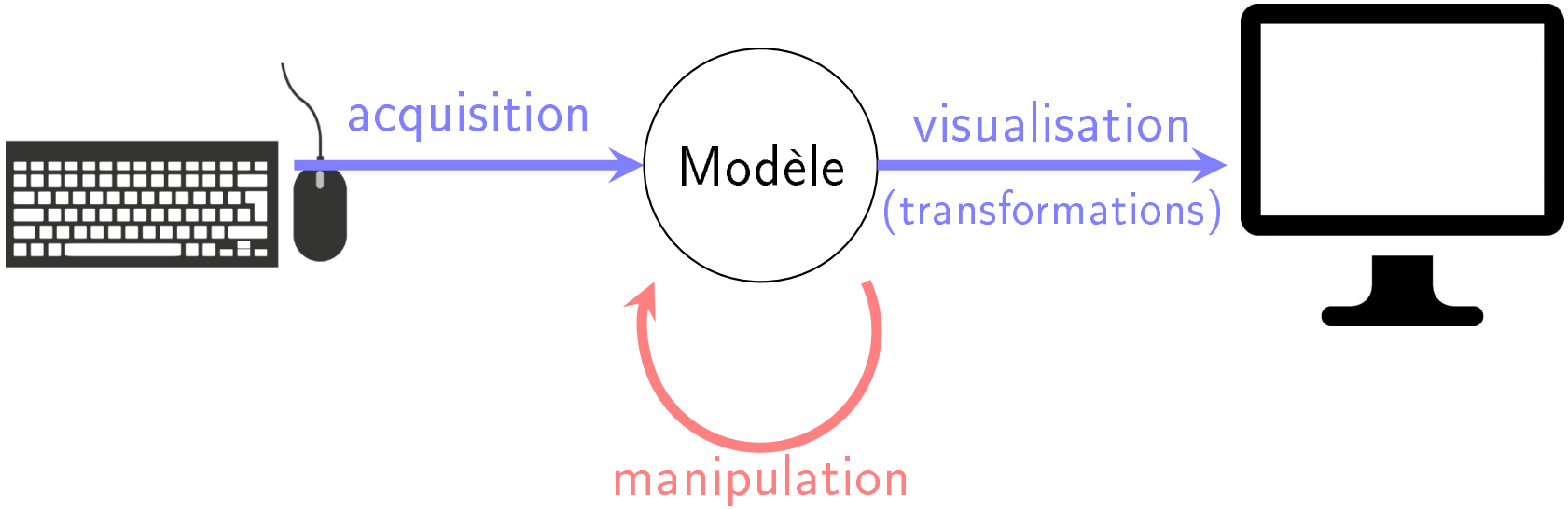
Quatre interfaces historiques : Presentation Manager (OS/2), Windows (DOS), Aqua (MacOS) et OSF/Motif (standard industriel Unix et norme).

Unix : "Gnome Foundation" (Sun, HP, Compaq et IBM ; RedHat, TurboLinux), KDE, CDE, OpenStep, ...

Avec l'émergence du standard HTML : interface sous forme de pages web (portabilité).

Les règles de conception des applications respectent la norme dans le modèle de présentation et celui d'acquisition.

Structure d'un système graphique interactif



Au niveau **materiel** : *réception* des données et *affichage* des images.

Au niveau **logiciel** :

- l'*application* gère les entrées, le modèle, et produit des commandes graphiques,
- le *modèle* représente les objets et leurs données,
- le *système graphique* est responsable de la production réelle de l'image.

Structure d'un système graphique
Le modèle

Une capture des données et des objets ainsi que de leurs relations.

Les objets sont représentés par :

- des données :
  - géométriques : polyèdres, courbes, . . ., ou
  - non géométriques : attributs, relation de voisinage, . . .
- des description procédurale (fractales par exemple),

Leurs interrelations sont diverses :

- interaction du programme (entrées),
- affichage (sorties),
- post-traitement non graphiques.

Structure d'un système graphique
L'affichage du modèle

Les modèles sont spécifiques aux applications et doivent être créés indépendamment de l'affichage.

Critères de sélection :

- géométriques,
- « traditionnels » (SGBD).

Les données extraites sont converties en données graphiques (primitives et attributs) par un processus de conversion :

1. traversée de la base de données,
2. conversion des données extraites à destination du système graphique.

Structure d'un système graphique
Gestion de l'interaction

Une boucle commandée par événement (machine à état fini) :

```
initialisation du systeme graphique

repeter
  attendre une action de l'utilisateur
  en fonction de l'action faire
    traiter la selection
    mise a jour du modele et de l'ecran
jusque termine

liberation des ressources graphiques
```

Structure d'un système graphique
Gestion de l'interaction

Dispositifs d'entrées :

- l'échantillonnage en mode asynchrone,
- l'attente de la génération d'un événement particulier en mode synchrone,
- la gestion d'une *file* d'événements en mode asynchrone.

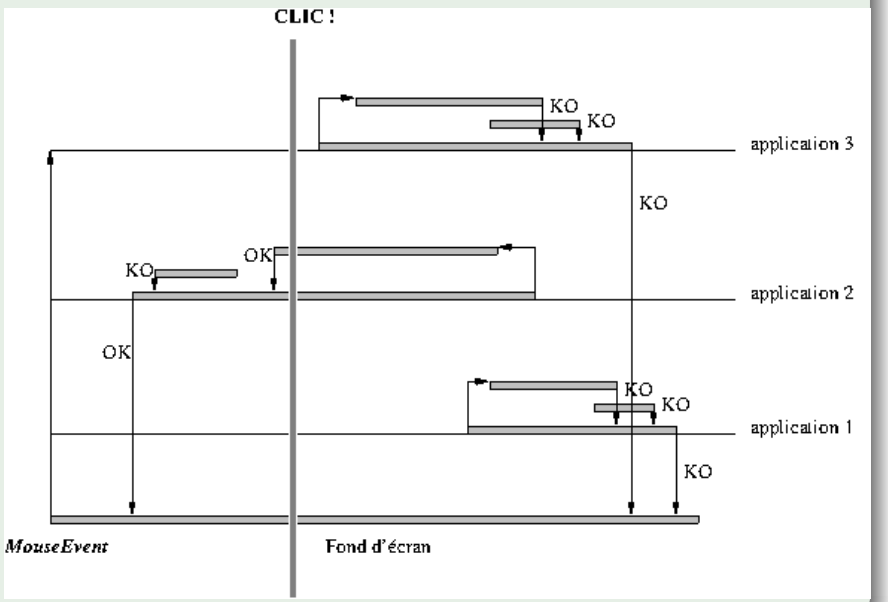
Réponses aux interactions de l'utilisateur :

- mise à jour de l'écran sans modification de la structure interne du modèle,
- mise à jour de l'écran depuis le modèle.

▶ aucune modification du modèle sous la responsabilité du système graphique.

Structure d'un système graphique
Gestion des évènement

clic-souris



algorithme

```
MouseEvent (event)

  si (controleur.aire(event)) alors

    pour fille dans file(fenetres_filles) faire
      si (fille.MouseEvent(event) = OK) alors
        # le traitement a eu lieu
        retourner OK
      finsi
    finpour

    traitement de l'evenement
    retourner OK

  sinon
    retourner KO

  finsi
```

Modèles d'architecture

Principe fondamental : séparation entre interface et noyau fonctionnel [2].

Différents modèles :

- de haut niveau :
  - modèle de Seeheim
  - modèle de l'arche
- de bas niveau :
  - Modèle-Vue-Contrôleur (MVC)
  - Présentation-Abstraction-Contrôle (PAC)

2. ensemble des services propres à un domaine d'application

Le noyau fonctionnel

Il doit fournir un minimum de services dans le cadre d'une interface [Fekete 1996] :

- notification : possibilité pour un module externe d'être prévenu lorsque l'état du noyau sémantique change,
- prévention des erreurs : possibilité de savoir si un appel de fonction est licite dans un certain contexte,
- annulation : possibilité de revenir à des états antérieurs du noyau sémantique, en évitant la simulation au niveau de l'interface.

Le noyau fonctionnel - Mise en œuvre

Notification

- modularité : le noyau fonctionnel ne connait pas le mode de présentation,
- pas d'attente active (*polling*),
- mécanismes de *callbacks* ou de variables actives (Tk),
- notification synchrone ou asynchrone ?

Prévention des erreurs

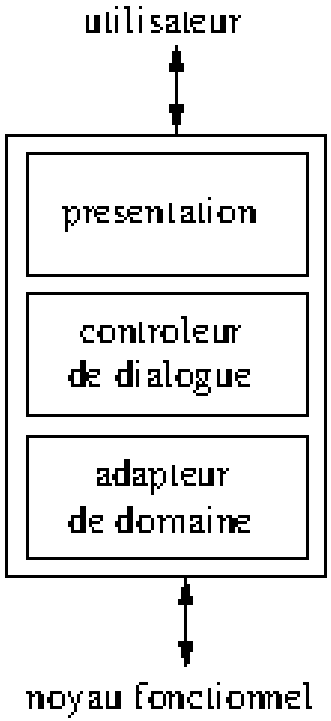
- pas avec des exceptions (c'est trop tard !),
- fonctions de test (accepté, refusé, ne sait pas).

Annulation

- mécanisme de *rollback* (annulation d'une transaction en BD), historique.

Le modèle de Seeheim

Patron de conception abstrait de haut niveau (Eurographics 1983).



Trois parties :

- présentation (lexical) : gère les E/S, affichage, interaction
- gestion du dialogue (syntaxique) : traitement et séquencement du dialogue
- adapteur du domaine (sémantique) : couche reliant les fonctions et données du noyau fonctionnel aux données et actions de l'interface.

Rôle historique d'organisation logicielle permettant de remplacer les interfaces textuelles en interfaces graphiques sans changer le noyau fonctionnel.

Modèle de l'arche

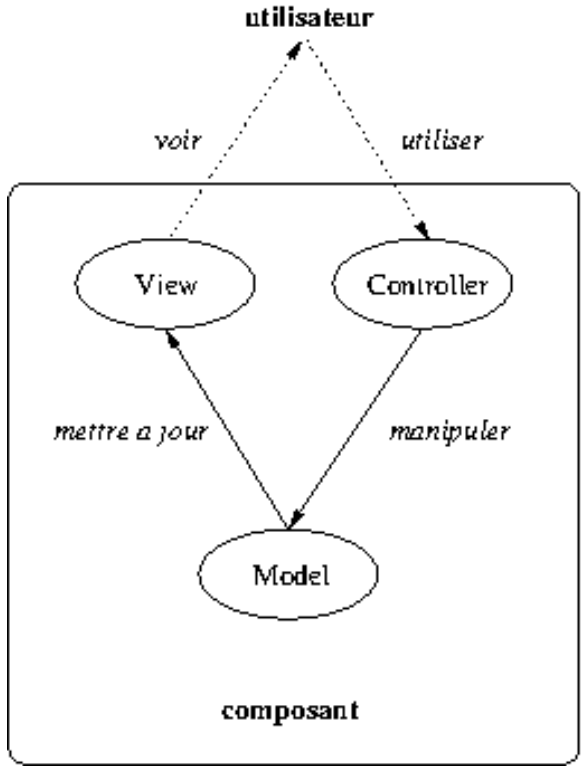
Patron abstrait de haut niveau évolution du précédent (1992) :

- noyau fonctionnel : données et traitements manipulés par le programme,
- adaptateur du noyau : médiation entre le noyau et la partie suivante,
- contrôleur de dialogue : gère le séquencement des tâches,
- présentation logique : médiation entre contrôleur et la partie suivante,
- présentation physique : périphériques et composants logiciels d'interaction.

Il vise essentiellement à analyser les architectures des applications et à réfléchir à leurs évolutions (graphique $\rightarrow$ vocale par exemple).

L'architecture MVC (*Model View Controller*)

Mis au point lors de la conception de l'environnement *Smalltalk* [Reenskaug 1979]. Il impose la séparation entre données, traitements et présentation.



L'architecture MVC

Principes :

- le *modèle* correspond aux données de l'application (structures et fonctions),
- la *vue* présente des informations à l'utilisateur à partir des données du modèle,
- le *contrôleur* se charge de l'interaction avec l'utilisateur.

Un client envoie une requête à l'application, celle-ci est analysée par le *contrôleur*, qui demande au *modèle* approprié d'effectuer les traitements, puis renvoie la *vue* adaptée si le modèle ne l'a pas déjà fait.

L'architecture MVC

Mise en œuvre

Modèle asymétrique :

- une paire contrôleur/vue est associée à un seul modèle,
- un modèle peut se voir associé plusieurs paires contrôleur/vue.

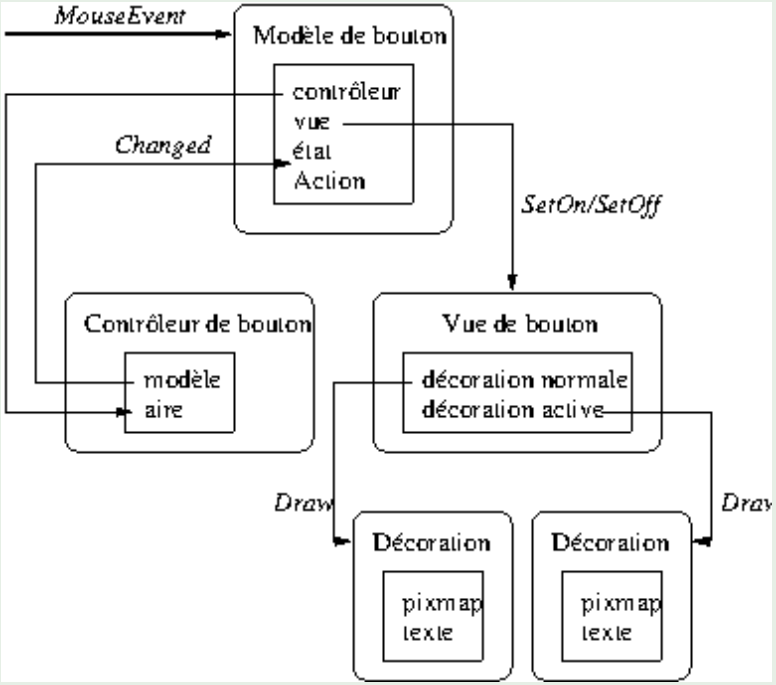
Listes dépendants et notification :

- les paires contrôleur/vue d'un modèle sont enregistrées dans une liste de "dépendants",
- lorsque le modèle change, tous les dépendants sont notifiés.

L'architecture MVC
Les boutons dans le modèle MVC

Trois aspects : présentation à l'écran (*vue*), gestion d'évènements (*contrôleur*) et coordination de la vue et du contrôleur (*modèle*).

modèle MVC



algorithme

```
MouseEvent (event)
  si (controleur.aire(event)) alors
    Changed()
    retourner OK
  sinon
    retourne KO
  finsi

Changed ()
  si (etat = ON) alors
    etat <- OFF
    vue.SetOff()
  sinon
    etat <- ON
    vue.SetOn()
    Action() // reference a une fonction
```

L'architecture MVC
Conclusion

- ⊕ vues multiples synchronisées,
- ⊕ vues et contrôleurs modulaires,
- ⊕ développement de composants réutilisables,
- ⊕ propagation naturelle du “look and feel”,
- ⊕ cohérence interne et externe des interfaces.

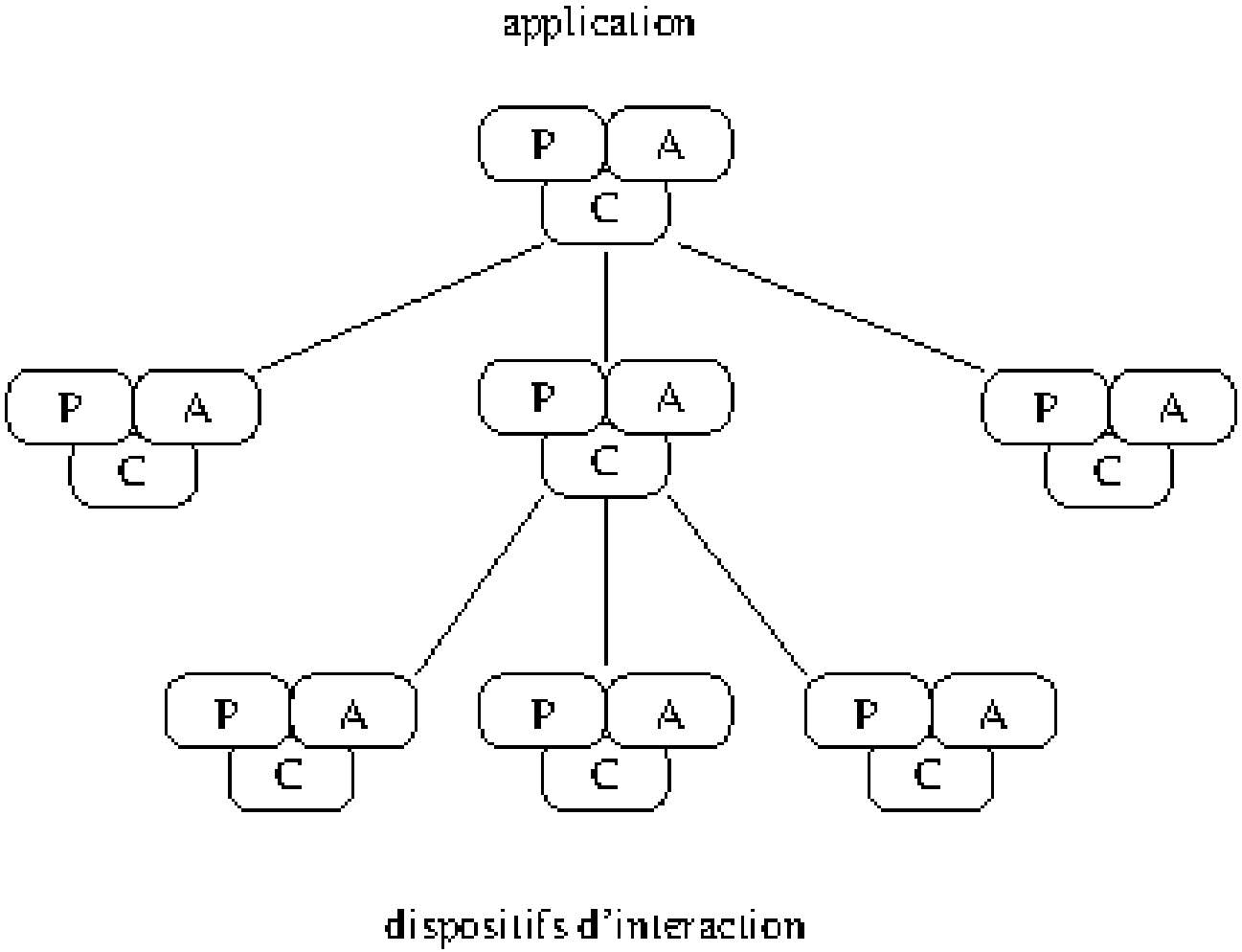
- ⊖ complexité de communication entre les composants

Modèle PAC

Décomposition récursive d'une application interactive par un modèle multi-agents [Coutaz 1980].
Le patron de conception organise un logiciel interactif en une hiérarchie de composants constitués de trois facettes :

- présentation (le V+C de MVC) : définit le comportement perceptible de l'agent, l'interaction avec l'utilisateur (IHM),
- abstraction (le M de MVC) : représente l'expertise de l'agent, gère les données à représenter (noyau fonctionnel),
- contrôle (pas d'équivalent dans MVC) : correspondance entre A et P (cohérence des représentations avec les données internes, conversion des actions de l'utilisateur en opérations du noyau fonctionnel) et gère les relations avec les autres agents PAC de la hiérarchie.

Structure hiérarchique



Comparaison MVC/PAC

Distinction entrées/sorties dans MVC mais pas dans PAC :

- MVC offre une meilleure souplesse/réutilisation possible de composants d'interaction sur des vues différentes, mais nécessite une communication accrue entre les objets *vue* et *contrôleur*.

Distinction entre *modèle* et communication entre la *vue* et le *modèle* :

- cette distinction n'est pas faite dans MVC, risque d'amalgame au niveau du *modèle*,
- dans PAC distinction entre *abstraction* et *contrôle*.

La norme CUA

CUA (*Common User Access*) est un standard pour les interfaces utilisateurs (IBM 1987).

Adopté par Microsoft Windows, Motif/CDE, GNOME, KDE, Java AWT et Swing, à des degrés divers.

Quelques principes généraux :

- toutes les opérations doivent pouvoir se faire à la souris ou au clavier,
- les menus s'activent/se désactivent par F10,
- une commande d'un menu est activée par la touche Alt suivie de la lettre appropriée,
- les options sont demandées dans une fenêtre indépendante, divisées en sections,
- toute application doit comporter un menu d'aide s'activant par la touche F1.

La norme CUA
La boîte de dialogue

Fenêtre spécialisée (une *fonction*, un *nom* ou titre, une *position*, une *taille*, un *état*, une *bordure*, ...), qui contient les contrôles (ou *widgets*) : boutons, barres de défilement, boîtes à cocher, ...

Distinction entre boîtes de dialogue modales (bloquantes) et non-modales.

La norme : pas plus de 30 contrôles actifs, disposés par famille, avec cadres et valeurs initiales, fenêtre non redimensionnable, positionnée au plus près de l'objet appelant sans le recouvrir, ...

Quatre boutons : `Ok`, `Apply`, `Cancel` et `Help`, alignement horizontal ou vertical, taille commune.

Le bouton `Cancel` peut être activé par la touche `ESC` et le bouton `Ok` peut l'être par la touche `RETURN`.

La norme CUA
La barre de menu

Accés permanent aux actions et aux commandes.

La norme :

- pas plus de huits options,
- options Fichier, Edition et Aide,
- un unique terme pour chaque option du menu,
- majuscules et minuscules pour chaque option du menu,
- ordonnez les options par priorité,
- groupez les par type de fonctionnalités.

Existence de : lignes séparatrices, codes émis depuis le clavier, commutateurs booléens, menus hiérarchiques.

Plan

Méthodes d'analyse et de conception

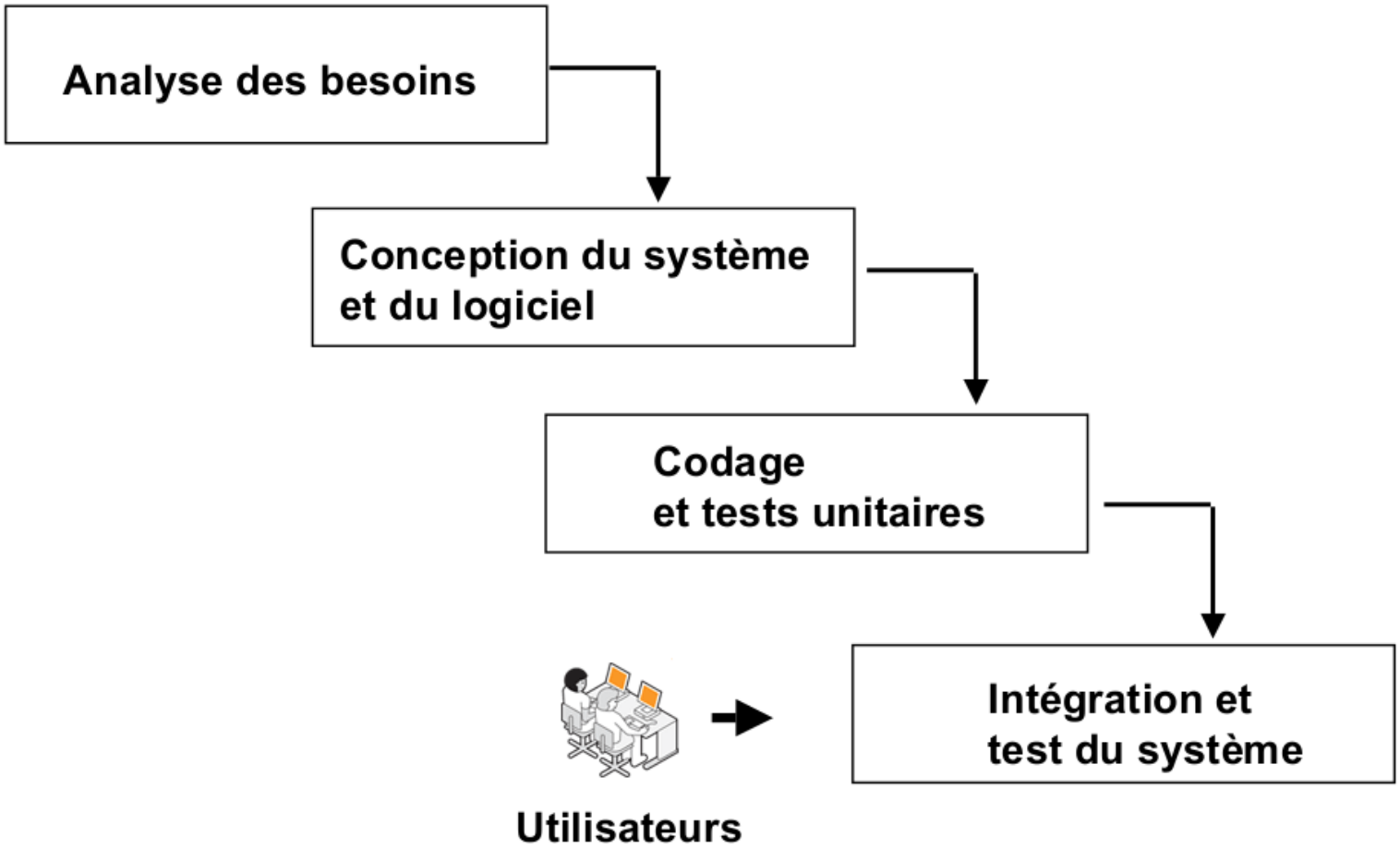
Quatre catégories : les méthodes ascendantes (de Blampré, ...), descendantes (ou systémiques : Merise, SADT, ...), orientées objets (OMT, ...) et orientée IHM (RAD : *Rapid Application Development*).

Avantages et inconvénients des méthodes RAD :

- ⊕ vision du produit en cours de développement, intégralité des solutions techniques, normalisation (communication, réalisation).
- ⊖ participation intense des utilisateurs, limitation des choix technologiques, non adaptées aux projets très techniques et/ou de longue durée.

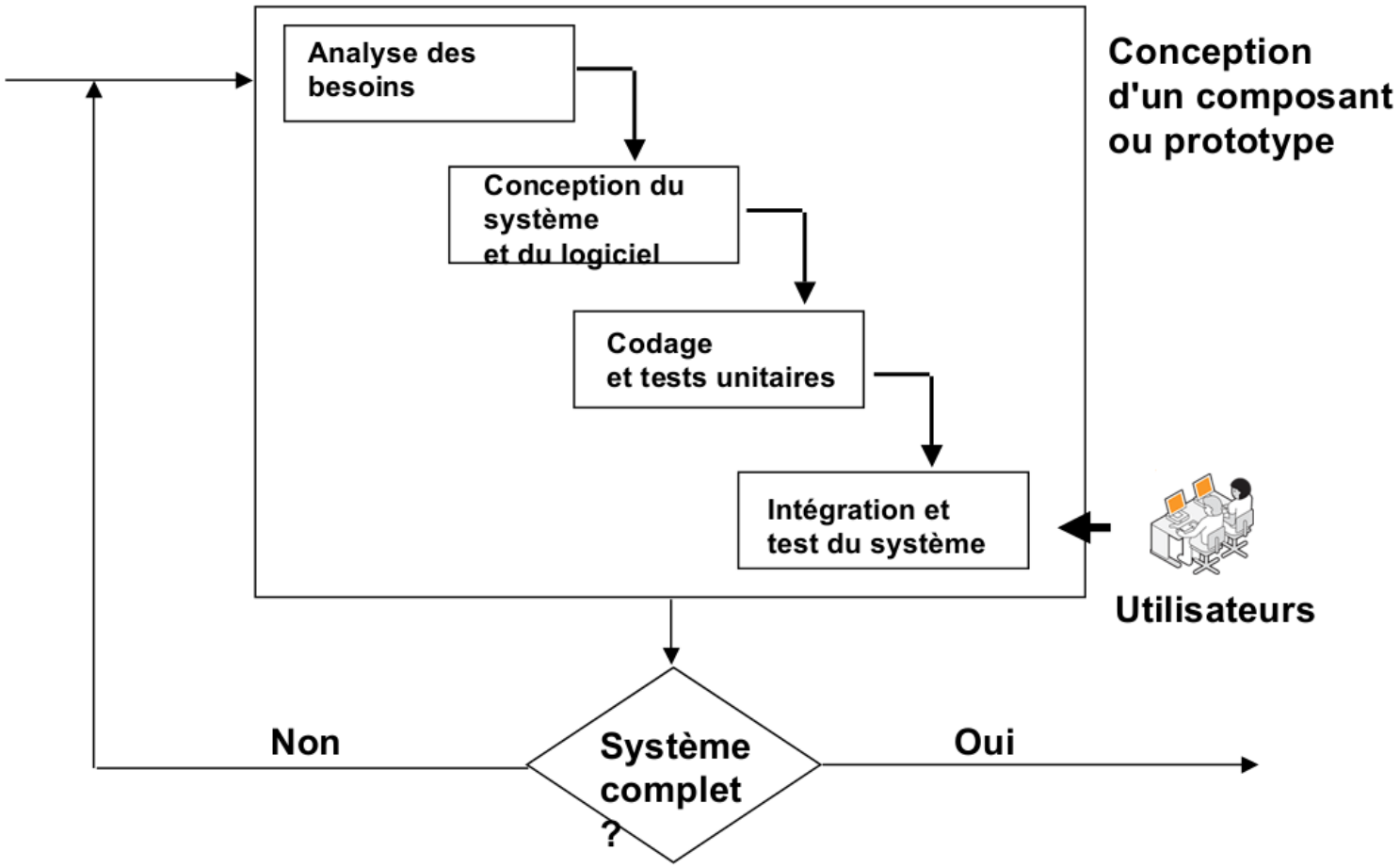
Méthodes d'analyse et de conception
Intégration des facteurs humains

La méthode classique : cycle de vie en cascade (*waterfall*).



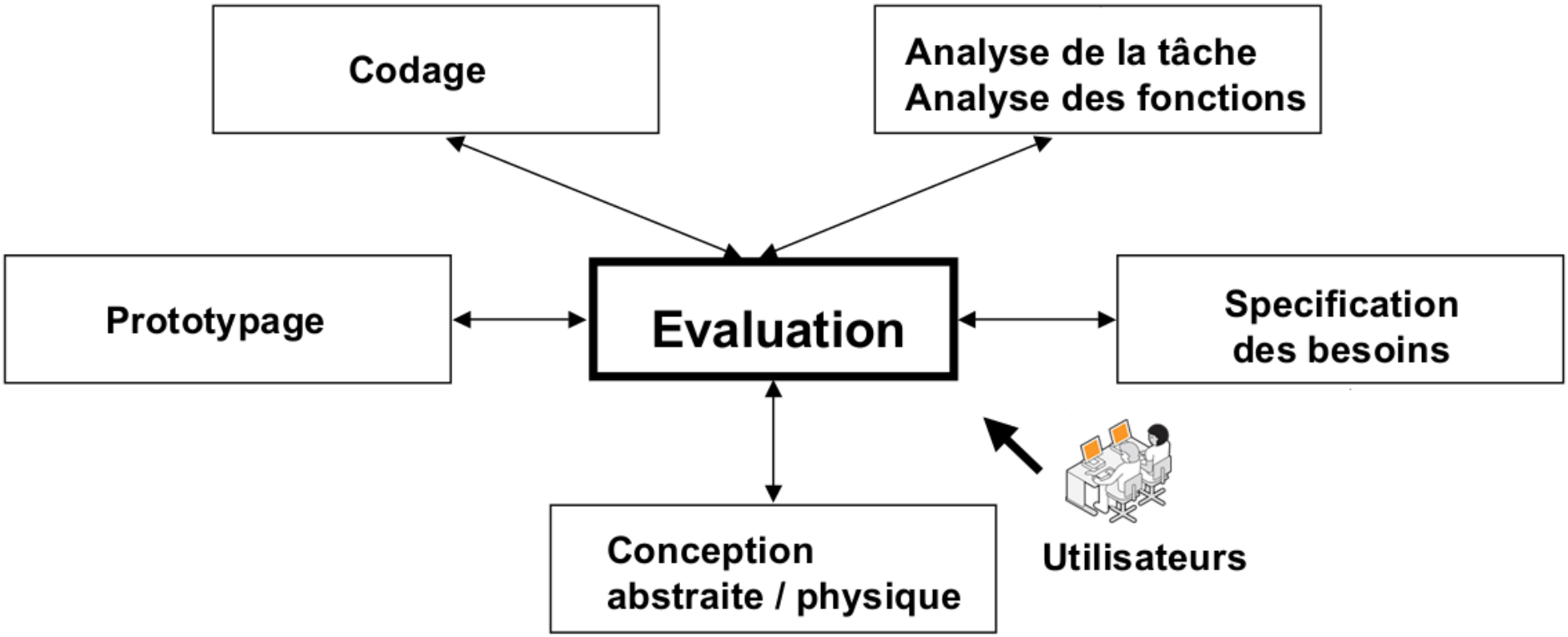
Méthodes d'analyse et de conception
Intégration des facteurs humains

Plus adaptée : l'approche incrémentale.



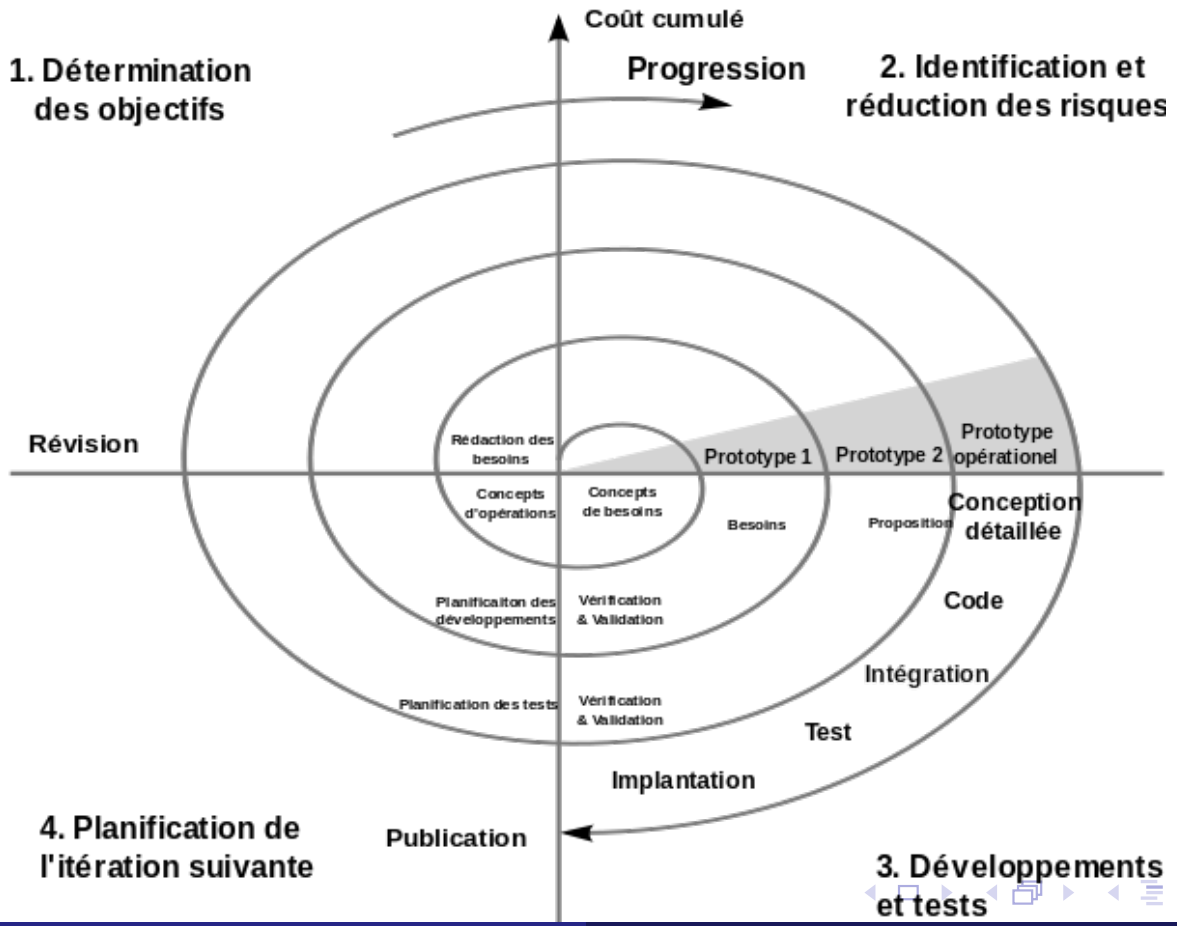
Méthodes d'analyse et de conception
Intégration des facteurs humains

Le cycle de vie en étoile (*star life cycle*).



Méthodes d'analyse et de conception
Modèle de Boehm (en spirale)

- incorpore la gestion du risque,
- durée constante aux cycles successifs du modèle.



Méthodes d'analyse et de conception
Comparaison

Séquentiel (S) vs incrémental (I), cinq critères :

- SI séparation entre MOA et MOE[3], risque de non adéquation de la solution informatisée aux attentes de la MOA,
- SI mauvaise conduite du changement (mais la forte implication des acteurs concernés représente un surcoût pour la MOA),
- SI pression sur la MOE,
- SI mauvaise visibilité sur les coûts et délais,
- SI mauvaise vision globale, architectures moins optimisées.

▶ ne pas être dogmatique (I = méthodes « agiles » ou « essais-erreurs » suivant les points de vue).

3. maîtrise d'ouvrage et maîtrise d'œuvre

Outils de développement

La programmation traditionnelle (L3G)

- ⊕ puissance de développement, optimisation des programmes, pérennité des langages.
- ⊖ difficulté de maîtrise du langage, difficulté de maîtrise de l'API, durée importante du développement, convient mal au prototypage.

La programmation visuelle

- ⊕ rapidité de développement d'une application, facilité d'utilisation, bonne adaptation au prototypage, orientation client-serveur, standardisation des programmes.
- ⊖ limitation des possibilités, programmes non optimisés, obligation d'utiliser un interprète (*runtime*) ou des bibliothèques dynamiques, langages de programmation propriétaires.

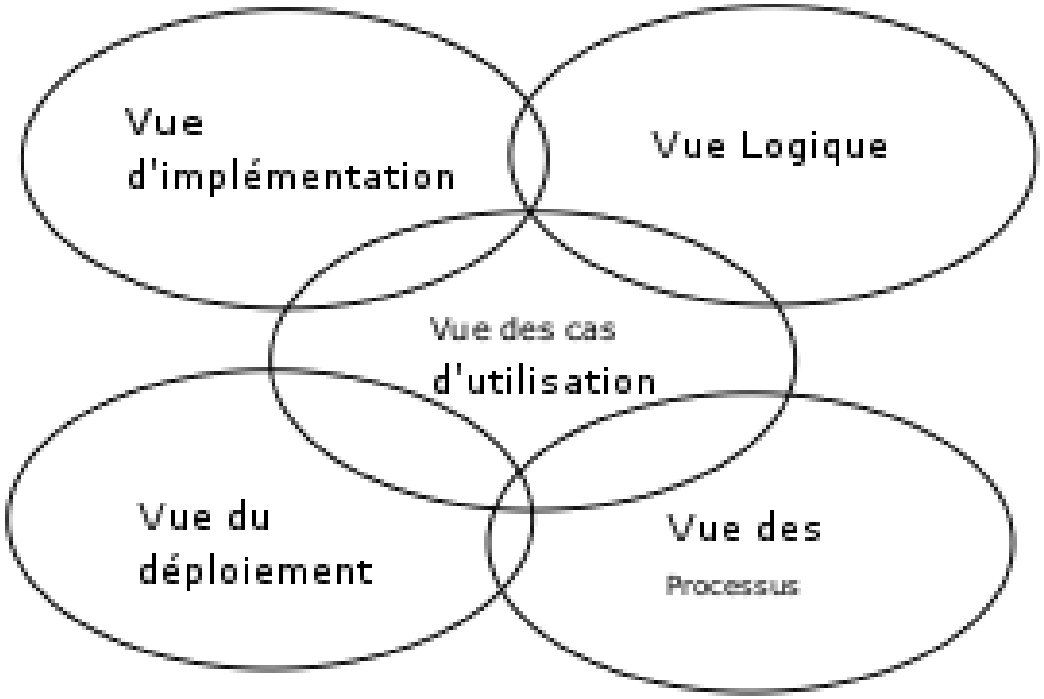
Conception - UML

Langage de modélisation graphique (*Unified Modeling Language*) à base de pictogrammes conçu en 1994 (UM).

- intègre le concept d'objet et les principes associés à l'objet notamment l'héritage, composition et transition d'état,
- formalisme de modélisation des spécifications directement compatible avec les équipes de développement,
- standard international.

14 diagrammes pour la modélisation d'un projet : statiques (classes, composantes, paquets, ...), de comportement (cas d'utilisation, activités, ...) et dynamiques (séquence, communication, ...).

Conception - UML



implémentation - dépendances, processus - aspect temporel, logique - comment, cas d'utilisation - qui, quoi, déploiement - où.

UML - diagrammes de cas d'utilisation

Première étape UML d'analyse d'un système.

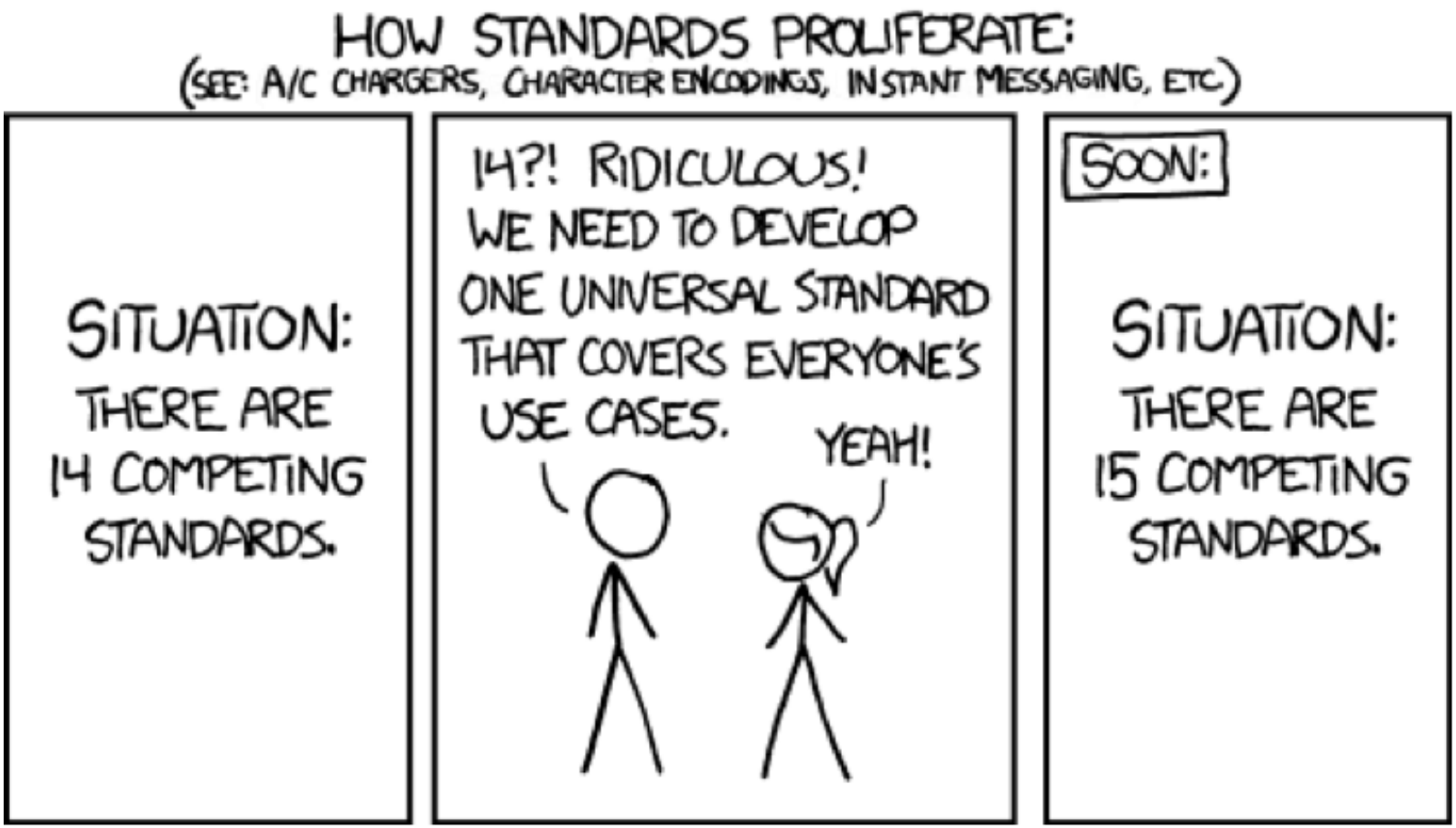
- toutes les fonctionnalités que doit fournir le système.

Rôle : moyen simple d'exprimer les besoins des utilisateurs, capture le comportement d'un système tel qu'un utilisateur le voit.

Pour élaborer les cas d'utilisation, il faut se fonder sur des *entretiens* avec les utilisateurs.

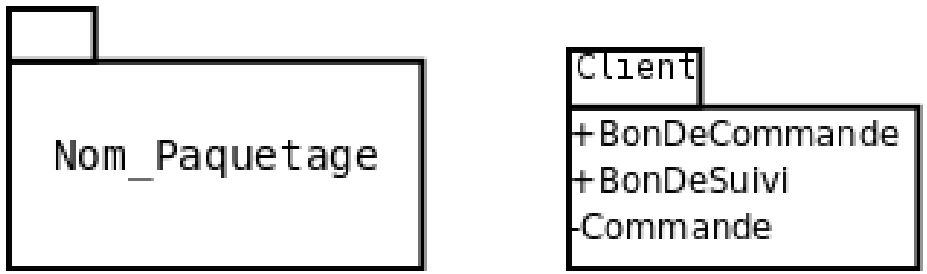
Par nature fonctionnel (et non objet) : difficulté pour trouver le bon niveau de détail (à éviter : décomposition fonctionnelle descendante hierarchique).

UML - diagrammes de cas d'utilisation



Use Case - Notions générales

Paquetages : regroupement d'éléments de modèle et de diagrammes (représenté par un dossier avec nom et contenu).



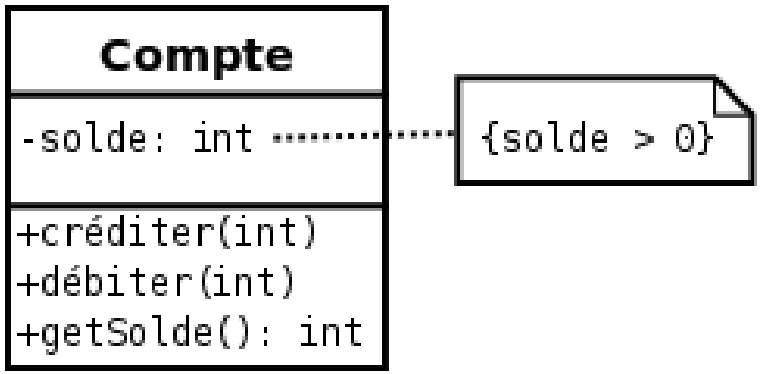
Espaces de noms : les éléments sont nommés de façon unique (syntaxe du séparateur des noms : :).

Classeurs : éléments de modélisation pouvant être instanciés (cas d'utilisation, acteurs, classes, interfaces, signaux, etc. ; représenté par un rectangle en traits pleins).

Use Case - Notions générales

Stéréotype : annotation s'appliquant sur un élément de modèle (représenté par une chaîne de caractères entre guillemets).

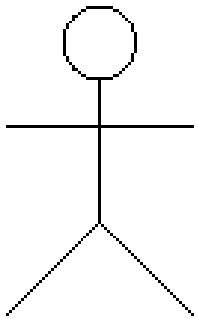
Note : information textuelle comme un commentaire, une méthode ou une contrainte (représentée par un rectangle dont l'angle supérieur droit est plié).



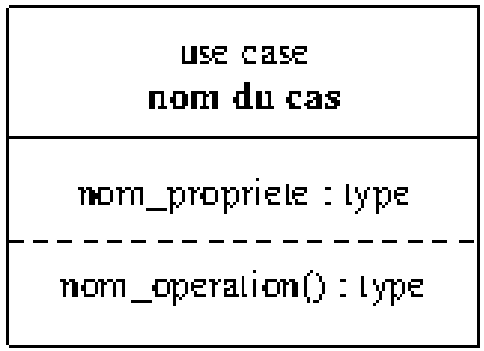
Use Case - Éléments

- Un acteur (utilisateur, processus, ...) interagit avec le système.
- Un cas d'utilisation (ellipse ou classeur stéréotypé « use case ») est une unité cohérente représentant une fonctionnalité visible de l'extérieur.

ACTEUR



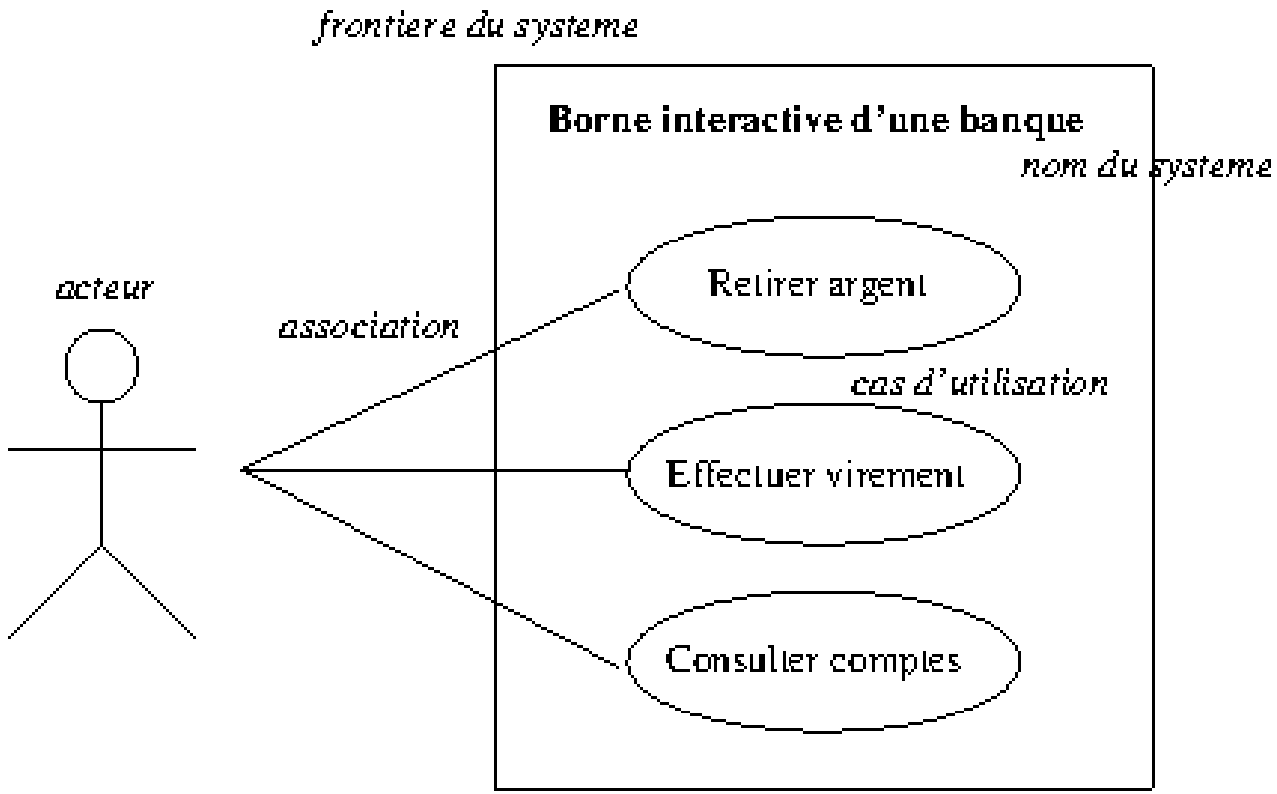
CAS D'UTILISATION



Use Case - Représentation

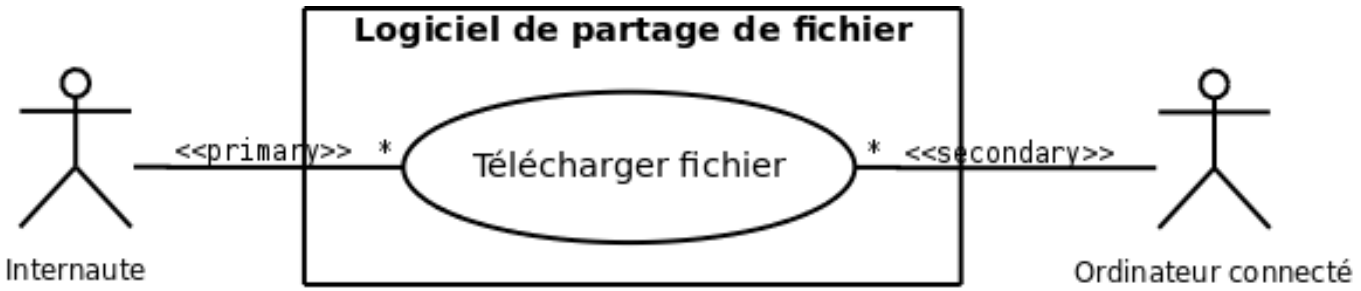
La frontière du système est représenté par un cadre.

Les acteurs sont à l'extérieur et les cas d'utilisation à l'intérieur.



Use Case - Relations entre acteurs et cas d'utilisation

- relation d'association : chemin de communication entre un acteur et un cas d'utilisation (trait continu).
- multiplicité : un acteur interagit plusieurs fois avec un cas d'utilisation (symboles `*`, `n`, `n..m`).
- acteurs : un acteur est principal (stéréotype « `primary` ») pour un cas d'utilisation si celui-ci rend service à cet acteur tandis qu'un acteur secondaire (« `secondary` ») est sollicité pour des informations complémentaires.
- cas d'utilisation interne : pas directement relié à un acteur.



Use Case - Relations entre cas d'utilisation

Deux types de relations

- les dépendances stéréotypées, surtout l'inclusion et l'extension.
  Une dépendance se représente par une flèche avec un trait pointillé. Si le cas A inclut ou étend le cas B, la flèche est dirigée de A vers B.
- la généralisation/spécialisation
  Une généralisation se représente par une flèche avec un trait plein dont la pointe est un triangle fermé désignant le cas le plus général.

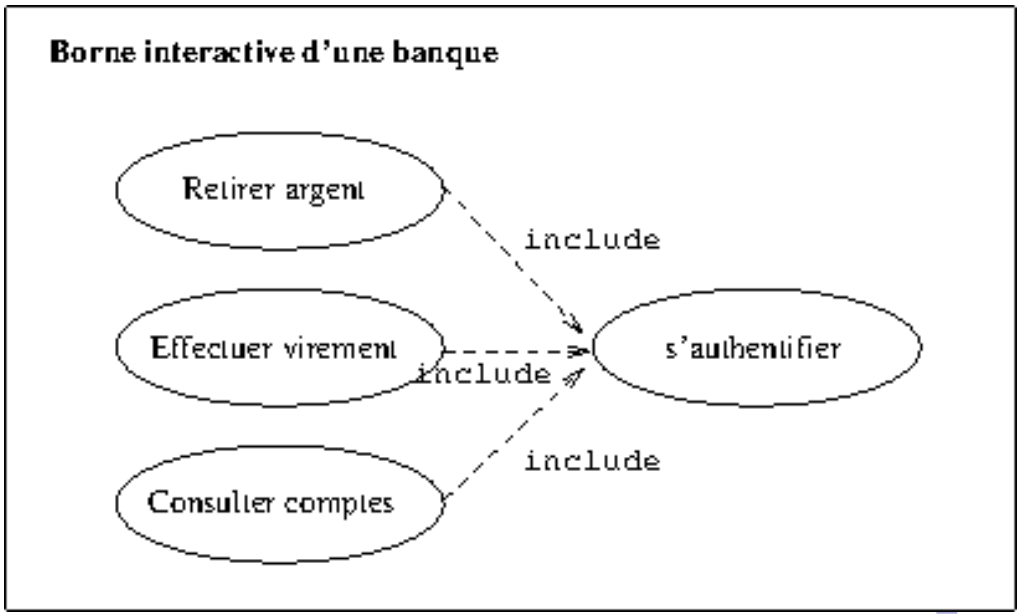
Use Case - Relations entre cas d'utilisation

Relation d'inclusion

Un cas A inclut un cas B si le comportement de A inclut celui de B (stéréotype « `include` »).

- permet de *factoriser* une partie d'un cas d'utilisation commun à d'autres cas d'utilisation.

Exemple : l'accès aux informations d'un compte bancaire inclut nécessairement une phase d'authentification.

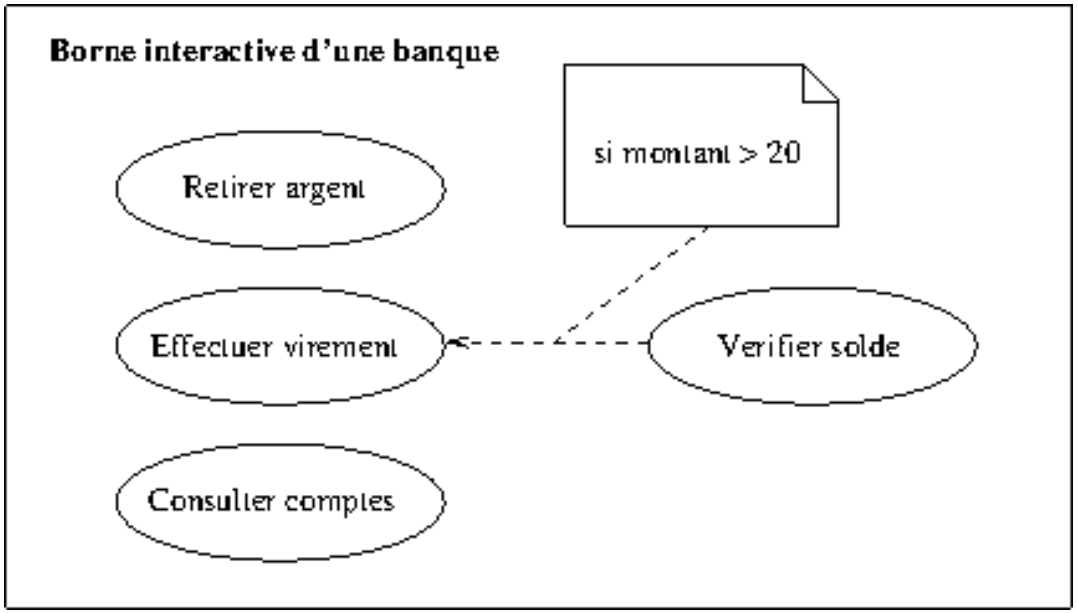


Use Case - Relations entre cas d'utilisation

Relation d'extension

Un cas A étend un cas B lorsque A peut être appelé au cours de l'exécution de B (stéréotype « `extend` »). Une extension est souvent soumise à condition (exprimée sous forme d'une note).

Exemple : la vérification du solde d'un compte intervient si la demande de retrait dépasse 20 euros.

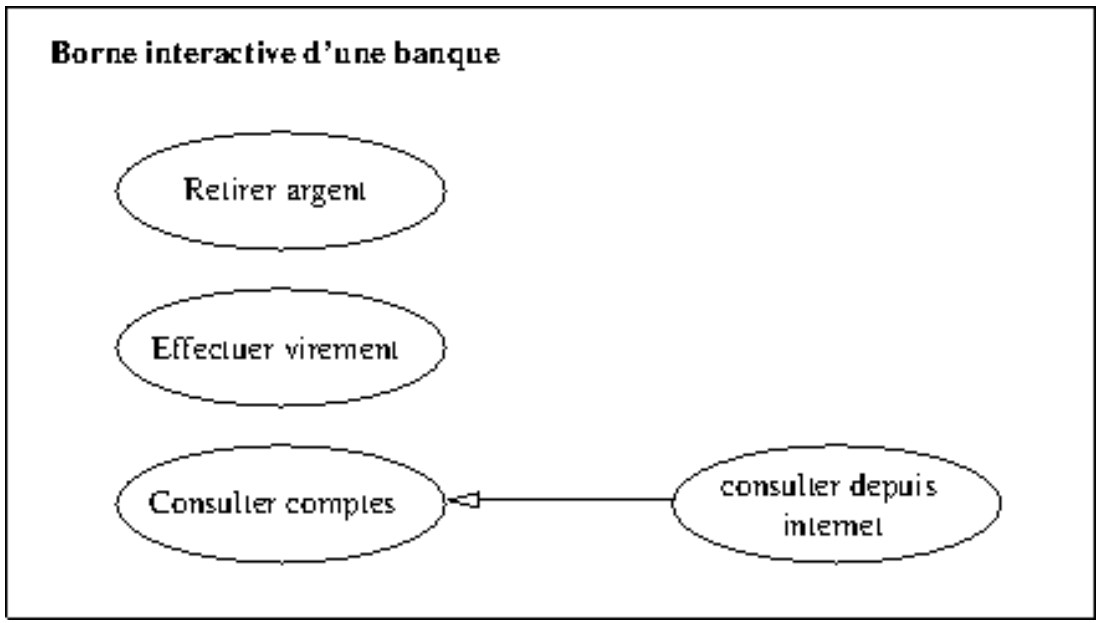


Use Case - Relations entre cas d'utilisation

Relation de généralisation/spécialisation

Un cas A est une généralisation d'un cas B si B est un cas particulier (une spécialisation) de A (concept d'héritage dans les langages orientés objet).

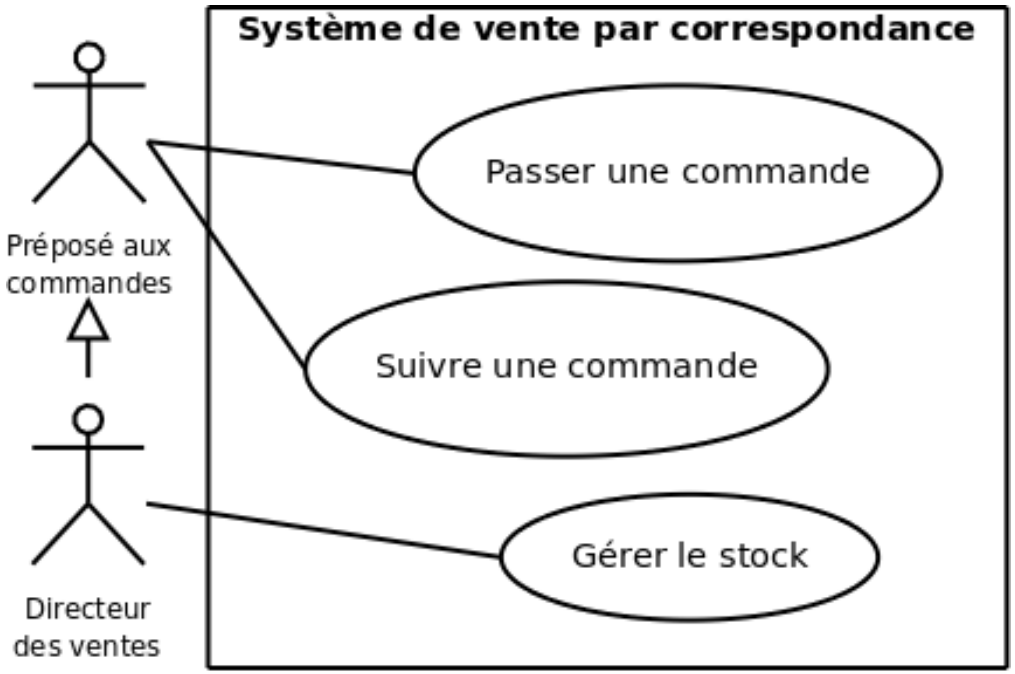
Exemple : la consultation d'un compte via Internet.



Use Case - Relation entre acteurs

La seule relation possible entre deux acteurs est la généralisation.

Un acteur A est une généralisation d'un acteur B si l'acteur A peut être remplacé par l'acteur B (symbole identique à la relation de généralisation).



Use Case - Description textuelle

Nom
Retrait d'espèces
Acteurs
Un client (primaire), le système informatique de la banque (secondaire)
Description
Le but de ce cas d'utilisation est, pour le client, de retirer de l'argent en espèces.
Scénario principal
le client introduit sa carte
le distributeur vérifie la validité de la carte auprès du réseau interbancaire
le client saisit son code secret
le distributeur vérifie le code
le client choisit l'opération "retrait d'espèces"
le client spécifie la somme à retirer
le distributeur demande au système informatique de débiter le compte
le distributeur rend la carte
le client prend la carte
le distributeur fournit les billets
le client prend les billets
Alternative : code incorrect
...
Pré-condition
Le client doit disposer d'une carte bancaire
Post-condition
Succès : le client dispose de sa carte et de l'argent en espèces
Échec : le client s'est vu saisir sa carte bancaire

UML - Diagramme d'activités

Représentation graphique du déroulement d'un cas d'utilisation, proche d'un diagramme d'états-transitions.

- modélisation du flot de contrôle et du flots de données (*workflow*) d'une activité à l'autre.

Représentation sous forme d'organigrammes ; à réserver à des opérations dont le comportement est complexe ou sensible.

Diagramme d'activités - Actions

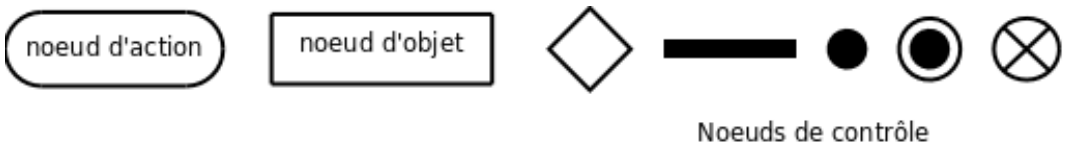
- appeler (*call operation*) : invocation d'une opération sur un objet de manière synchrone ou asynchrone,
- comportement (*call behavior*) : invoque directement une activité (plutôt qu'une opération),
- envoyer (*send*) : crée un message et le transmet (appel asynchrone) à un objet cible, où elle peut déclencher un comportement,
- accepter événement (*accept event*) : bloque l'exécution en cours jusqu'à la réception du type d'événement spécifié (signal asynchrone),
- accepter appel (*accept call*) : variante de l'action accept event pour les appels synchrones,
- répondre (*reply*) : transmission d'un message en réponse à la réception d'une action de type accept call,
- créer (*create*) : permet d'instancier un objet,
- détruire (*destroy*) : permet de détruire un objet,
- lever exception (*raise exception*) : permet de lever explicitement une exception.

Diagramme d'activités - Groupe et noeud

Une activité est un comportement (*behavior*) dont le flot d'exécution est modélisé par des nœuds reliés par des arcs (transitions).

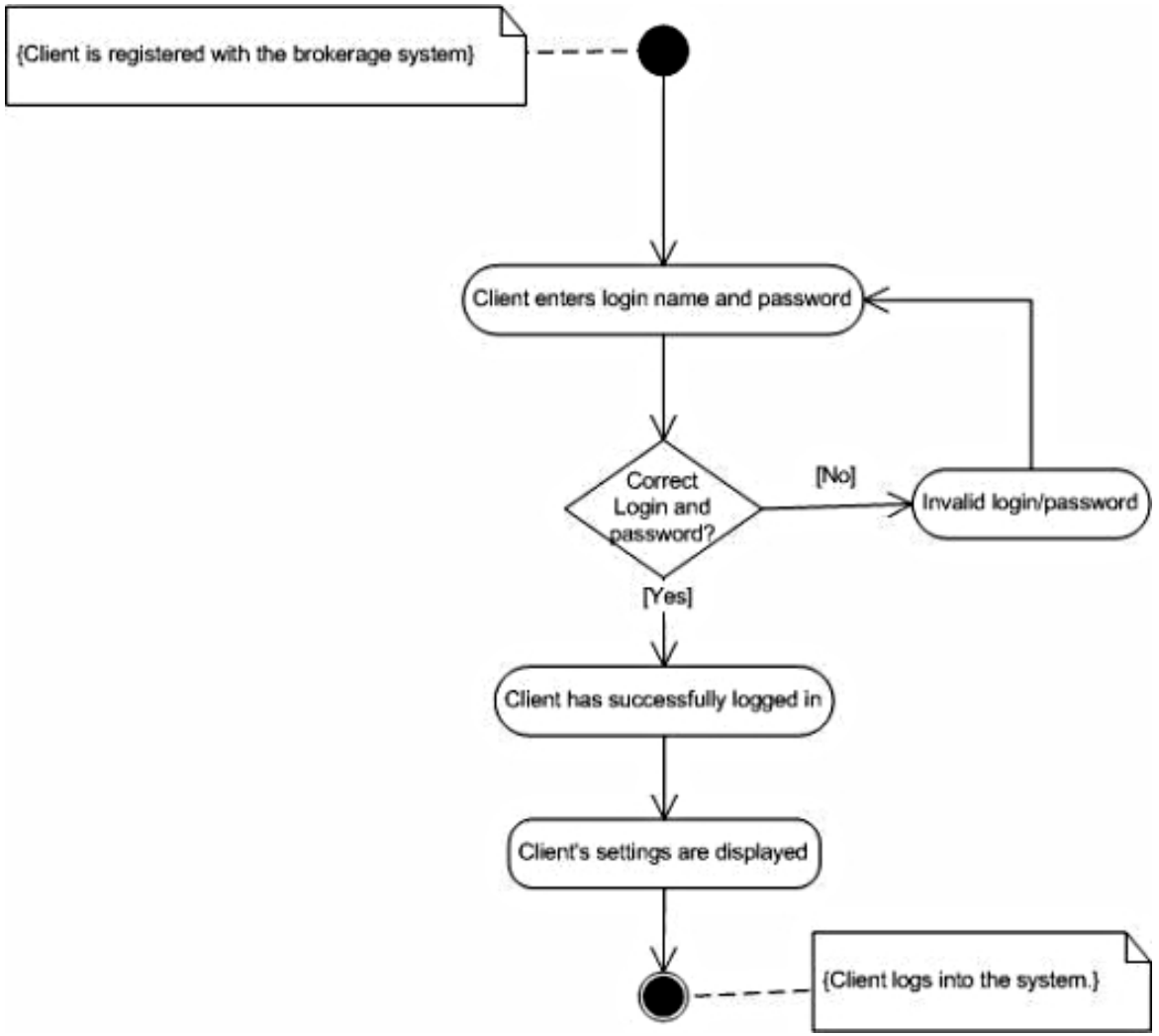
Un nœud d'activité est un type d'élément abstrait permettant de représenter les étapes le long du flot d'une activité. Il existe trois familles de nœuds d'activités :

- les nœuds d'exécutions (*executable node*),
- les nœuds objets (*object node*),
- et les nœuds de contrôle (*control nodes*).



Noeuds de contrôle

Diagramme d'activités



Conception dirigée par les buts

Elle doit répondre aux questions suivantes :

- qui sont les utilisateurs ?
- que cherchent-ils à faire ?
- comment cela est-il conçu ?
- à quelles expériences font-ils appel ?
- comment les utilisateurs interagissent ?
- quel est le comportement de mon produit ?
- quelle est l'aspect de mon produit ?
- comment organiser les fonctionnalités efficacement ?
- quelle ergonomie pour masquer la technologie ?
- comment résoudre les problèmes rencontrés ?
- comment se présenter aux utilisateurs débutants ?
- comment aider les utilisateurs inexpérimentés ?
- comment fournir profondeur et puissance aux experts ?

Conception dirigée par les buts
Scénarios

Scénario vs use case : dans les deux cas description de l'interaction de l'utilisateur, mais deux objectifs différents :

- use case : description exhaustive des fonctionnalités d'un système, catalogue des tâches utilisateurs
  - ▶ le comportement n'est pas vraiment pris en compte.
- scénario : définition du comportement du produit vis à vis d'archétypes, présentation et priorité des fonctionnalités
  - ▶ définit le « quoi » avant le « comment » (pas de spécifications).

Trois types de scénarios : contextuels, des points clés et de validation.

Scénario contextuel

Utilisation d'un PDA par Alice, agente immobilière, dont les buts sont un équilibre entre travail et vie personnelle, conclure des affaires, donner l'impression de ne s'occuper que du client du moment.

1. au matin, Alice vérifie son courriel sur son PDA, celui-ci ayant un écran confortable et une liaison réseau rapide cela est plus pratique que d'allumer son PC, d'autant qu'elle prépare un sandwich pour sa fille,
2. son dernier client, Bob, souhaite visiter une maison l'après-midi, le PDA a connaissance de ses coordonnées (action simple depuis son email),
3. au téléphone avec Bob, elle commute sur le haut-parleur car elle souhaite consulter son agenda : en ajoutant un rdv, le PDA le fait au nom de Bob car il est en communication ; Alice ajoute l'adresse du bien dans l'agenda,

Scénario contextuel

4. au bureau, elle prend d'autres rdv qui sont automatiquement ajoutés à l'agenda de l'agence,
5. la journée passe rapidement, le PDA prévient Alice qu'il ne reste que 15 mn avant le rdv avec Bob, en consultant son PDA il y a toutes les informations associées à Bob (courriels, mémos, mesg tél., log des appels) ; en demandant à téléphoner le PDA propose Bob car il sait que le rdv est proche, Alice prévient d'un retard,
6. Alice connait l'adresse du bien mais elle n'y est jamais allée ; son PDA lui indique sa position par rapport au bien,
7. le PDA d'Alice sonne durant la visite (qu'elle a basculé sur répondeur pour ne pas être dérangée) mais le PDA reconnaît un appel de sa fille et bascule en mode normal,

Scénario contextuel

8. sa fille a raté le bus, Alice téléphone à son mari et tombe sur son répondeur : elle laisse un message. Cinq minutes plus tard, son PDA émet un son de réception d'un message propre à son mari : il s'en charge.

- identifier les besoins par objet, action et contexte ; exemple : appeler (action) une personne (objet) depuis un rdv (contexte).
- utiliser une interface « magique », oublier les contraintes technologiques.

Conception dirigée par les buts

Approche *top-down* afin de ne pas être influencé par les solutions techniques.

1. entrées (forme, posture, input),
2. données et fonctions,
3. groupes et hiérarchies de fonctions,
4. esquisse de l'interaction,
5. scénarios des points clés (itération sur point 4),
6. scénarios de validation.

Entrées :

- forme : écran LCD, smartphone, condition d'éclairage, borne dans un espace publique,
- posture : degré d'attention de l'utilisateur et réaction du système,
- input : clavier, souris, keypad, écran tactile, voix, contrôleur de jeux, contrôleur distant, hardware dédié.

Conception dirigée par les buts

Données et fonctions :

- données : sujets principaux utilisés (photos, courriels, ...), attributs de chaque objet, relation entre chaque donnée (association, inclusion, ...),
- fonctions : opérations possibles sur les données, éléments de l'interface.

Grouper données et fonctions en unités fonctionnelles, déterminer leur hiérarchie, organiser par contenant (écrans, fenêtres, frames). Points à considérer :

- quels éléments nécessitent une large part de l'écran ?
- quels éléments contiennent d'autres éléments ?
- comment organiser les conteneurs pour optimiser la tâche ?
- quels éléments sont utilisés ensemble et suivant quelle séquence ?
- quels éléments doivent être connus à chaque décision ?
- quel(s) type(s) d'interaction ?

Conception dirigée par les buts

Esquisser le schéma d'interaction : phase « rectangles » de haut niveau, ne pas rentrer dans les détails, processus itératif à faire en petit groupe collaboratif avec des outils adaptés.

Scénarios des points clés : cheminement à travers l'interface pour les tâches les plus fréquentes.

Scénarios de validation :

- alternatifs, exemple : Alice souhaite répondre par email à son client Bob,
- nécessaires mais non fréquents, exemple : mise à jour, initialisation, configuration, nettoyage, procédure d'effacement en cas de revente,
- exceptionnels, priorité des développeurs car sources d'instabilité mais à ne pas traiter dans la phase de conception, exemple : Alice veut mémoriser un contact dont le nom est identique à un autre.

Concevoir le comportement

Interaction basée sur ce qui est attendu d'un service.

- Intéressé : se souvenir de vos goûts.
- Respectueux : des suggestions, pas des ordres.
- Prévenant : supplément d'informations en relation avec la demande initiale, exemple : une demande d'impression précise rarement que le niveau de papier ne suffit pas, qu'il y a 10 demandes avant vous et qu'il y a une autre imprimante de libre.
- De bon sens : ne pas fournir de fonctions inappropriées à une place inappropriée.
- Discret : certaines informations ne doivent être conservées qu'explicitement (numéro de carte, identifiant, compte, mot de passe, ...).

Concevoir le comportement

- Prévoyant : profiter des temps morts pour devancer les demandes, exemple : précharger les pages des url présentent dans une page web.
- Consciencieux : faire plus que le strict nécessaire, exemple : lors d'une copie signaler les doublons, proposer de renommer, ...
- Pudique : ne pas tenir sans cesse l'utilisateur de ses succès et de ses échecs, la plupart du temps constat inverse (messages d'erreur, de confirmation, notification, ...).
- Perspicace : fournir la bonne information au bon moment, mémorisation fine des préférences, exemple : enregistrer l'utilisation du plein écran après plusieurs sessions.
- Confiant : éviter les messages systématiques de confirmation, exemple : effacer un document mais se préparer à le restaurer.

Concevoir le comportement

- Calme : laisser le choix (magasin), ne pas poser sans cesse des questions (oral), exemple : préférence d'une langue.
- Serviable : réparer les erreurs, exemple : retour sur un formulaire (vide !) après validation.
- Conciliant : la plupart des systèmes ne connaissent que deux états, conforme ou non conforme, tout système manuel connait au moins un troisième état en attente, exemple : mettre de coté une tâche à laquelle il manque une information,
- Responsable : l'utilisateur n'a pas à subir certains impératifs techniques, exemple : une demande d'impression annulée côté client est souvent mal traitée côté serveur d'impression.
- Protecteur : éviter les erreurs gênantes, exemple : courriel personnel à une liste, ...