Interface Homme Machine

1. Introduction
2. Définitions
3. Critères ergonomiques
4. Les étapes du processus de développement et IHM
5. Modèles de tâches
6. Architecture logicielle des systèmes interactifs
7. Les formalismes de validation des DHMs (Dialogues Homme Machine)

6. Architecture logicielle des systèmes interactifs

Les modèles d'architecture interactives ont été développés pour modéliser les applications interactives. Ils permettent de décomposer l'application en plusieurs modules tels que chacun a un rôle qui lui est attribué. Cette décomposition peut suivre trois procédés distincts:

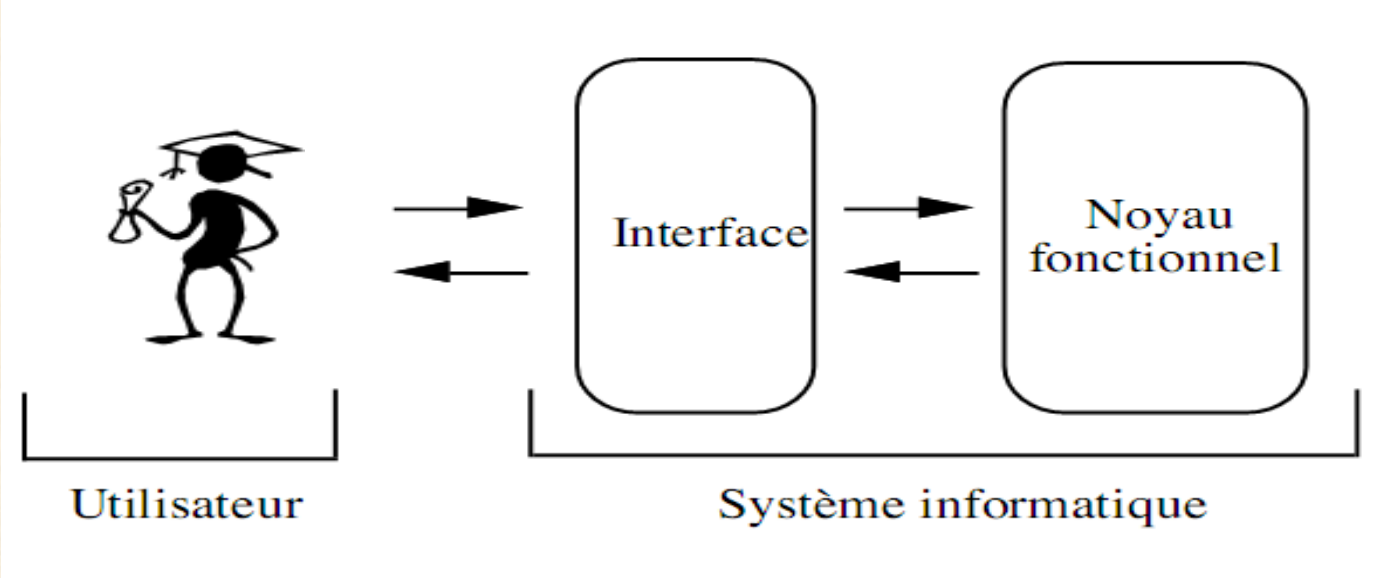
- ✓ une décomposition de l'application dans son ensemble (**modèles globaux**),
- ✓ une décomposition bi-axiale (**modèle multi-agents**) et
- ✓ une décomposition combinant les avantages des deux premières décompositions (**modèles hybrides**).

6.1) Principe de base:

Bien que les modèles d'architecture se diffèrent, ils répondent tous au même principe de base de la figure au dessous:

- ✓ la distinction entre les services d'une application et
- ✓ les fonctions assurant l'interaction avec l'utilisateur.

L'application, appelée aussi **Noyau Fonctionnel (NF),** regroupe les fonctionnalités du domaine et les opérations qui leur sont applicables. L'**interface** présente à l'utilisateur les tâches et fonctions et de lui permettre de les manipuler selon un enchaînement défini par le modèle de tâches (Exemple: CTT).



6.1) Principe de base (Cont.):

Par conséquent, un modèle d'architecture pour systèmes interactifs doit :

- ✓ définir une décomposition modulaire des fonctionnalités (i.e., stratégie de répartition des services du système interactif)
- ✓ indiquer la séparation entre les services du noyau fonctionnel et ceux de l'interface.

But : définir des modules, leurs composants, leurs relations et leurs propriétés, pour spécifier une architecture.

6.2) Les étapes de développement d'une architecture logicielle

Une architecture désigne une structure du logiciel selon un point de vue. Comme pour tout objet conçu, plusieurs structures selon des points de vue différents sont possibles. La figure suivante distingue trois étapes principales à partir des requis fonctionnels d'un système interactif :

1**) Décomposition fonctionnelle**

Décomposition en unités simples suivant un modèle de référence (décomposition normalisée pour un problème commun).

2**) Décomposition modulaire**

Associer les fonctions à des modules (groupes de fonctions, regroupe des aspects communs à des fonctions)

2 règles à suivre: minimiser les dépendances et couverture fonctionnelle (complète, pertinente)

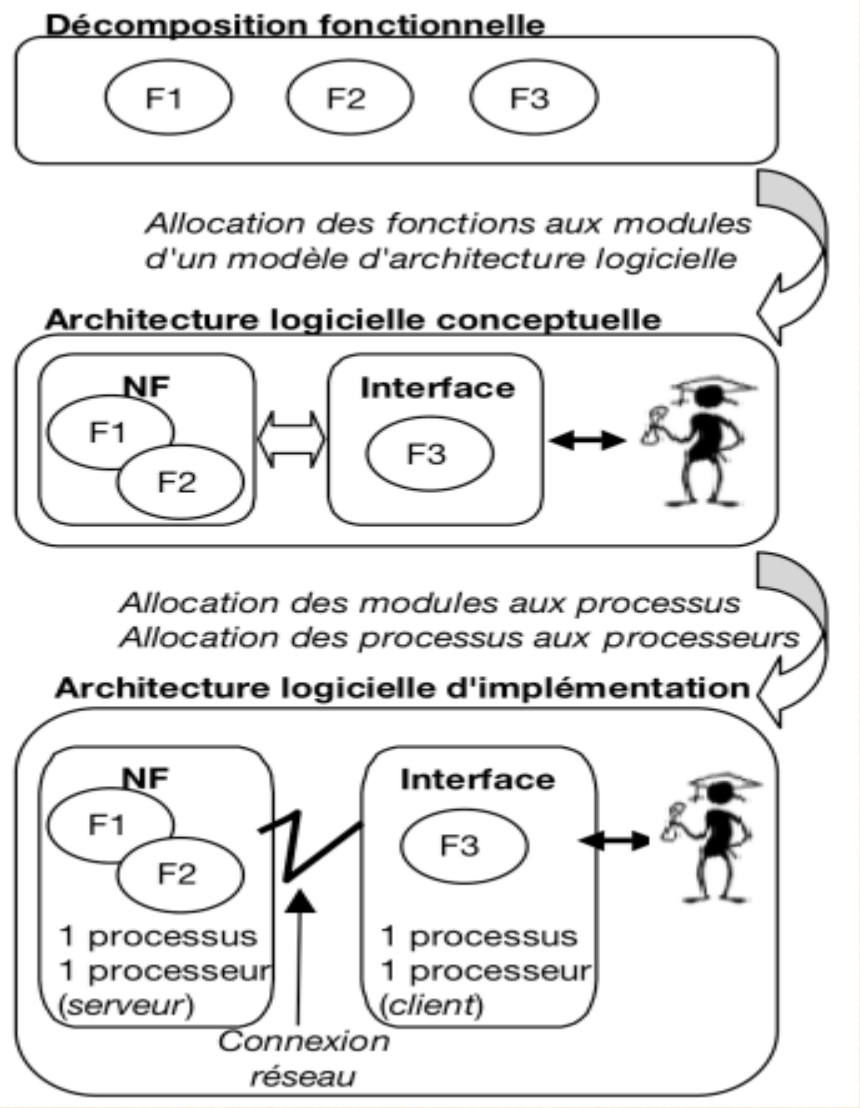
3**) Décomposition processus/Processeur**

Affecter les processus logiques aux ressources de calculs physiques.

6.2) Les étapes de développement d'une architecture logicielle

Exemple :

Processus de conception d'une architecture : exemple d'application du principe de base de la figure au slide 3



6.3) Modèles de référence pour les systèmes interactifs

On a trois classes de modèles d'architecture : **les modèles globaux** qui fournissent des structures fonctionnelles à gros grain (i.e., globales, structure interne des modules non détaillée), **les modèles multi-agents** qui visent une décomposition fonctionnelle plus fine et **les modèles hybrides**.

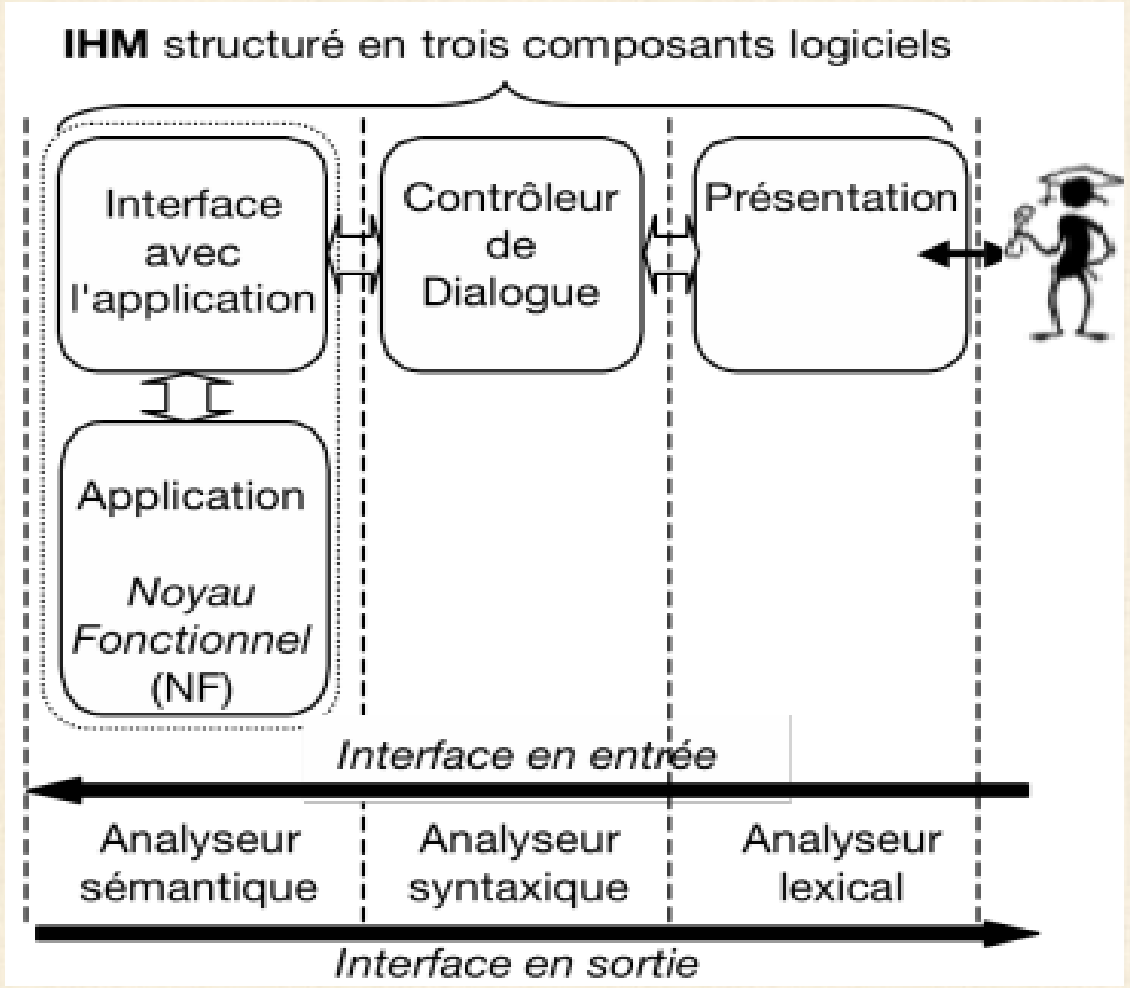
1. Modèles Globaux: (Seeheim et Arch)

Les modèles globaux ou fondamentaux structurent l'application en la considérant dans son ensemble. Ils permettent d'établir l'architecture d'une application en fonction des différentes facettes jouées par le code.

1.1. Le Modèle Seeheim :

Le premier modèle d'architecture ayant été défini est le modèle de Seeheim (figure au dessous). Celui-ci trouve son origine dans la nécessité de remplacer les commandes textuelles par les interfaces graphiques sans avoir à redéfinir le noyau fonctionnel des applications existantes.

Il décompose l'application en trois modules (en plus du noyau fonctionnel) : ***l'adaptateur*** ou *l'interface* avec le noyau fonctionnel, ***la présentation***, et enfin ***le dialogue,*** qui permet de lier les deux modules précédents en maintenant leur cohérence.



– **La Présentation** qui est la partie dédiée à l'interface graphique de l'application, elle est chargée de rendre perceptible l'état pertinent des concepts du domaine (la propriété d'observabilité) et d'en permettre la manipulation par l'utilisateur,

Elle est chargée aussi de la gestion purement lexicale de l'application. Son rôle est de :

- ✓ gérer la présentation et l'affichage des entités manipulées. Elle doit donc répondre aux critères d'ergonomie et de facilité d'apprentissage (utilisabilité) d'une application.
- ✓ gérer les événements physiques d'entrée/sortie. Elle est chargée de “convertir les événements de bas niveau en événements de haut niveau” ou plus simplement de “traduire les informations entre le format du monde réel extérieur et celui du monde informatique intérieur”

– Le Contrôleur de Dialogue (CD) sert de pont d'indirection (médiateur) entre le NF (application) et la Présentation (utilisateur). En tant qu'intermédiaire, le Contrôleur de Dialogue gère les relations entre le NF et la Présentation, c.-à-d., il contrôle les échanges au niveau syntaxique, et gère la dynamique du dialogue.

- L'interface avec l'application qui correspond à la vue qu'a le contrôleur de dialogue sur le noyau fonctionnel (le Noyau Fonctionnel ou applicatif regroupe les concepts et fonctions du domaine). Elle a pour rôle :

- ✓ de relier le noyau fonctionnel et le module de contrôle du dialogue en faisant appel aux procédures de l'application.
- ✓ de convertir les informations issues du contrôleur de dialogue en concepts manipulables par l'application. L'interface de l'application doit également savoir comment sont structurés les services de l'application et comment les utiliser. En d'autres termes, elle doit connaître la sémantique de l'application.

Les features du Modèle Seeheim :

- ✓ Ce modèle a permis la définition du vocabulaire qui est utilisé dans l'ensemble des autres modèles. C'est-à-dire, il est le premier qui a mis en valeur **le contrôleur de dialogue** par l'intermédiaire de la séparation d'une application en trois modules.

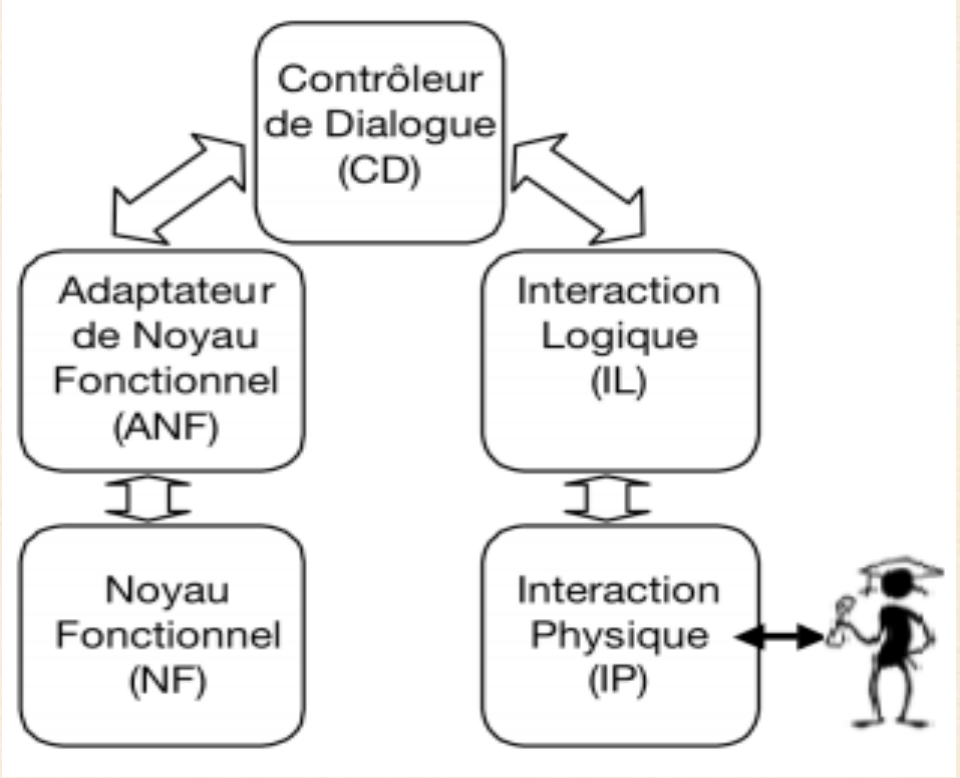
- ✓ Ce modèle a pour but de distinguer la gestion du dialogue de la sémantique de l'application. Il est composé de trois modules qui sont greffés sur le noyau applicatif.

- ✓ Chacun de ces modules est distinct des autres et possède des rôles bien déterminés. Cependant, il ne précise pas comment réaliser les différents modules ni comment les liens entre les différents modules doivent être implémentés.

1.2. Le Modèle Arch :

Afin de tenir compte des évolutions techniques (l'apparition des boîtes à outils comme JDK, API windows etc.), le modèle Seeheim a été complété en donnant naissance au modèle ARCH (Figure en bas).

- ✓ Ce modèle est composé des mêmes modules que Seeheim auxquels est ajouté un autre module et le noyau fonctionnel qui est intégré au modèle d'architecture.
- ✓ Ce module ajouté est la « boîte à outils » (Toolkit) sur laquelle repose l'implémentation de la présentation



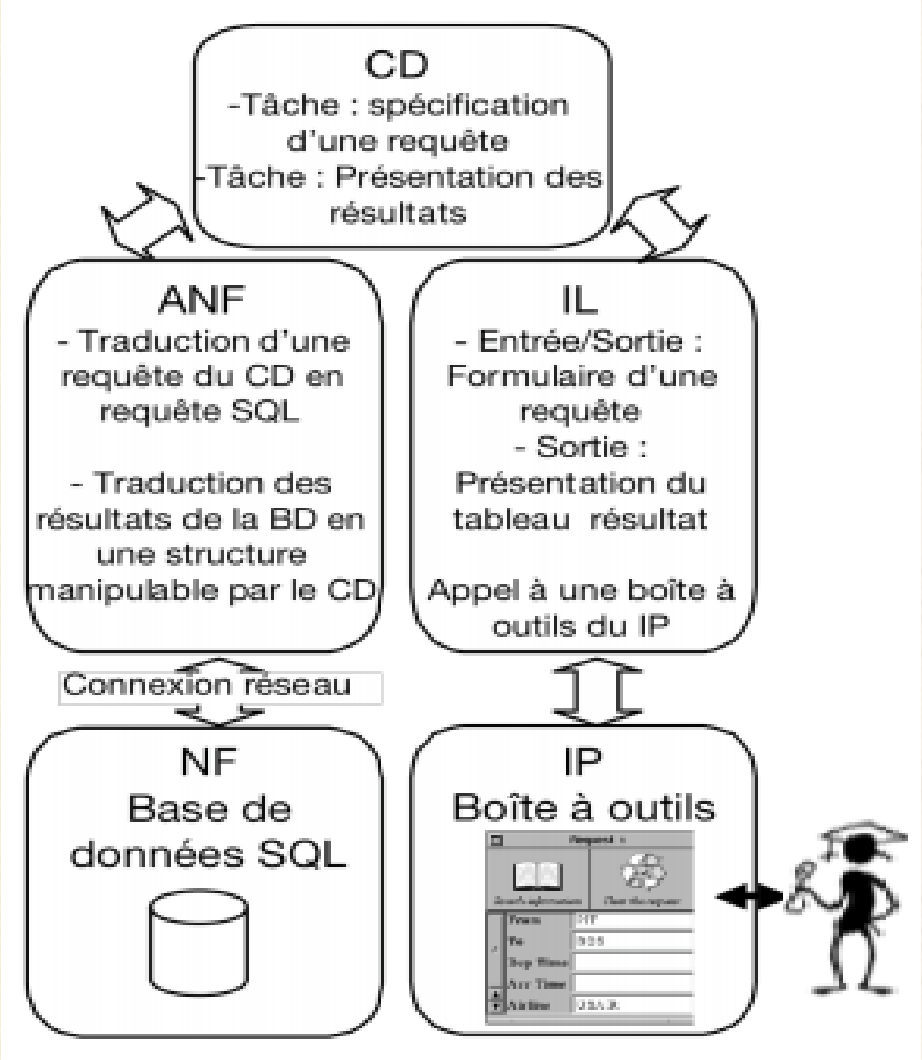
1.2. Le Modèle Arch (Cont):

- ✓ Les pieds de l'Arch constituent les composants imposés par la réalité:
  - le ***noyau fonctionnel*** réalise les différentes tâches du domaine (Services du système),
  - le composant ***d'interaction physique***, en contact direct avec l'utilisateur, est mis en œuvre au moyen des objets d'interaction d'une boîte à outils.
- ✓ Le ***contrôleur de dialogue*** gère l'enchaînement des tâches ainsi que les liens entre les objets regroupés dans les deux composants voisins : *l'adaptateur de noyau fonctionnel* et le composant *d'interaction logique*.
- ✓ ***L'adaptateur de noyau fonctionnel*** sert à ajuster les différences de modélisation des objets conceptuels entre le noyau fonctionnel et le contrôleur de dialogue.
- ✓ Le composant ***d'interaction logique*** permet au ***contrôleur de dialogue*** de s'affranchir du fonctionnement de l'interface GUI basée sur une boîte à outils au niveau de l'interaction physique. Le composant ***d'interaction logique*** peut se voir comme une boîte à outils virtuelle qui implémente des objets d'interaction logique concrétisés par une Interface GUI.

Exemple du Modèle Arch :

La ***figure suivante*** présente un exemple d'architecture ***Arch*** pour un système interactif simple d'interrogation d'une base de données distante :

- ✓ l'utilisateur spécifie une requête en s'aidant d'un formulaire et il obtient les résultats à l'écran sous la forme d'un tableau.



Bien que le modèle **ARCH** permet de prendre en compte les avancées dans le domaine des IHMs, il a les mêmes inconvénients que le modèle de **Seeheim** :

- ✓ il ne précise ni la structure interne des modules, ni les méthodes d'échanges entre eux.

2. Modèles multi-agents

Un Agent ?

Un agent est un système de traitement de l'information : il possède des ***récepteurs*** et des ***émetteurs*** pour acquérir et produire des ***événements*** ;

- ✓ Il dispose d'une ***mémoire*** pour mémoriser son état.
- ✓ Il comprend un ***processeur*** spécialisé dans le traitement d'une ou plusieurs ***classes d'événements***.
- ✓ Le résultat d'un traitement se traduit généralement par un ***changement d'état*** de l'agent et par l'émission de ***nouveaux événements*** en envoyant des ***messages*** aux autres agents.
- ✓ Deux types d'agents :
  - ***Cognitif (intelligent)*** pour les tâches complexes et
  - ***réactif*** : agent simple, pas intelligent c'est l'ensemble qui résout.

2. Modèles multi-agents (Cont.)

Les modèles multi-agents procèdent en réalisant deux décompositions successives de l'application suivant deux axes:

i) Tout d'abord une décomposition en agents (d'où le nom), c'est-à-dire que l'application globale se compose d'un ensemble de *packs* correspondant chacun à un concept (tâche ou activité).

ii) Ces packs, qui se communiquent, sont décomposés en trois modules chacun remplissant un rôle déterminé. Les définitions de ces modules sont très proches de celles des modules des modèles globaux. C'est à dire, pour le modèle PAC: la **Présentation**, **l'Abstraction** et le **Contrôle**, et pour le modèle MVC: le **Modèle**, la **Vue** et le **Contrôle**.

2. Modèles multi-agents (Cont.)

De ce fait ces modèles structurent un système interactif en un ensemble d'agents spécialisés réactifs. Ils se caractérisent par une organisation fortement modulaire, des traitements exécutés en parallèle et une communication par événements (i.e., par envoi de messages).

Les modèles multi-agents sont apparus avec les langages de programmation objets POO et sont aujourd'hui les modèles sous-jacents des langages utilisés pour la conception d'interface.

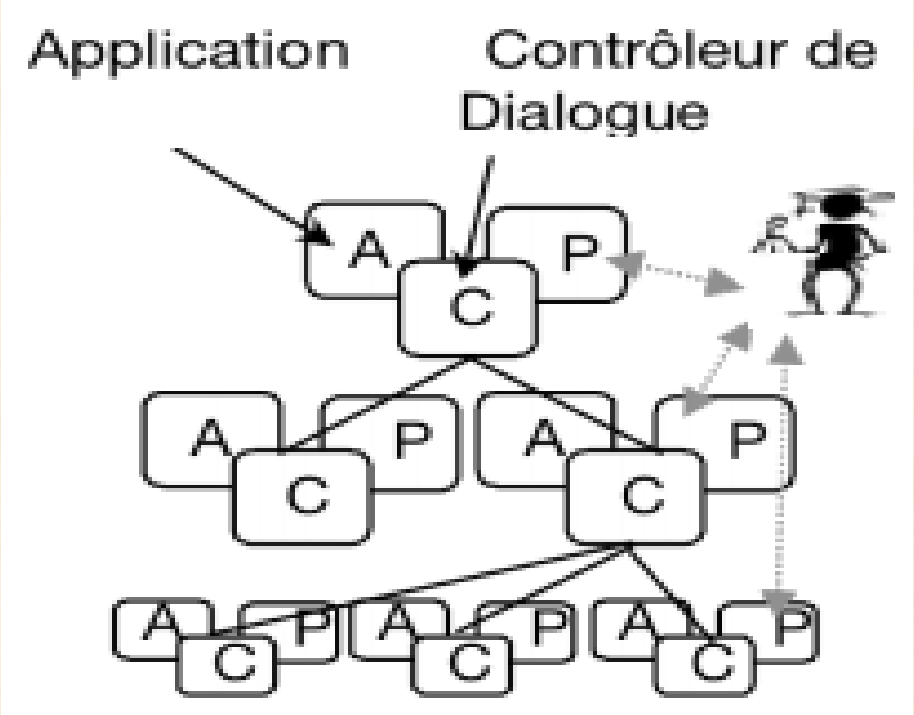
Par exemple les objets de la librairie Java/Swing sont implémentés via le modèle d'architecture ***Modèle-Vue-Contrôleur (MVC).***

2.1. Le modèle PAC

Comme le montre la Figure ci-dessous, le **modèle PAC**, proposé par Coutaz, structure récursivement un système sous forme d'une hiérarchie d'agents.

La hiérarchie permet d'exprimer les relations entre les agents et reflète un continuum de niveaux d'abstraction depuis l'application jusqu'aux éléments fins de l'interaction.

l'*Abstraction* du niveau le plus haut (racine de la hiérarchie) correspond à la notion d'*Application* du modèle Seeheim et le *Contrôle* du niveau le plus haut remplit des fonctions similaires au Contrôleur de dialogue du même modèle.



Un ***agent PAC*** définit une compétence à un niveau d'abstraction donné. C'est un acteur à trois facettes :

- **la *Présentation*** définit le comportement perceptible de l'agent, c'est-à-dire les aspects visuels, sonores... ainsi que la réception des actions des utilisateurs.

- **l'Abstraction** représente son expertise, elle définit les fonctionnalités de l'agent (indépendamment de l'interaction). Cette partie correspond au noyau fonctionnel du modèle globale Seeheim.

- et **le Contrôleur** a un double rôle :

  (1) il sert de lien entre les facettes *Présentation* et *Abstraction* de l'agent.

  (2) il gère les relations (i.e., Communications) avec d'autres agents PAC de la hiérarchie.

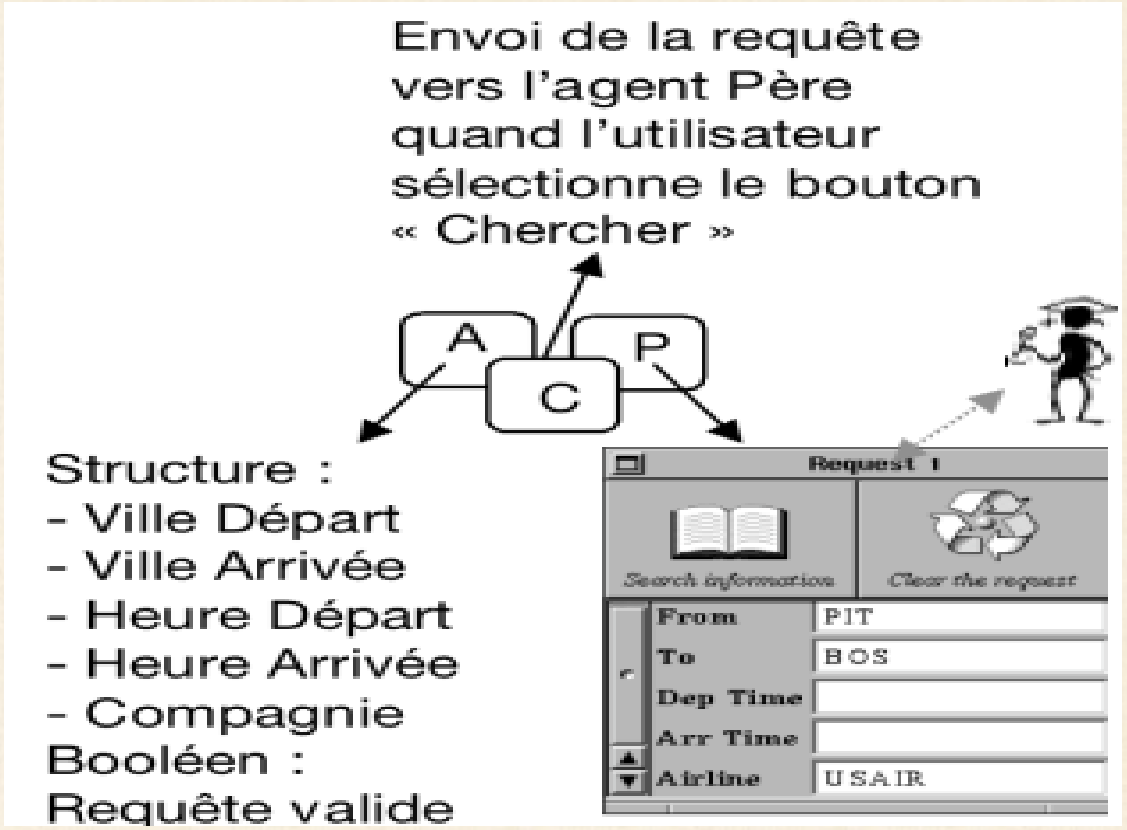
2.1. Le modèle PAC (Cont.)

L'ensemble des agents PAC d'une application communiquent via leurs Contrôles, qui sont donc également en charge de la cohérence entre les différents agents de l'application. C'est pourquoi la partie contrôle est indispensable de tout agent PAC alors que les deux autres modules peuvent être absents. Il faut remarquer aussi qu'un objet père (interface) hérite des Présentations de ses fils (objets) et non l'inverse.

L'exemple suivant présente un agent PAC élémentaire, feuille de la hiérarchie : la **Présentation** sait dessiner un formulaire avec deux boutons et reçoit les événements qui traduisent les actions de l'utilisateur ;

L'Abstraction, avec ses valeurs d'état constitue un modèle ou structure de l'état d'une requête et **le Contrôle** effectue le pont entre les deux facettes en gérant la correspondance des structures abstraites et concrètes. Par exemple, lorsque l'utilisateur a saisi au moins les **villes de départ** et **d'arrivée** dans le formulaire, l'Abstraction déclare que la requête est valide et on informe la Présentation via le Contrôle pour activer le bouton Chercher.

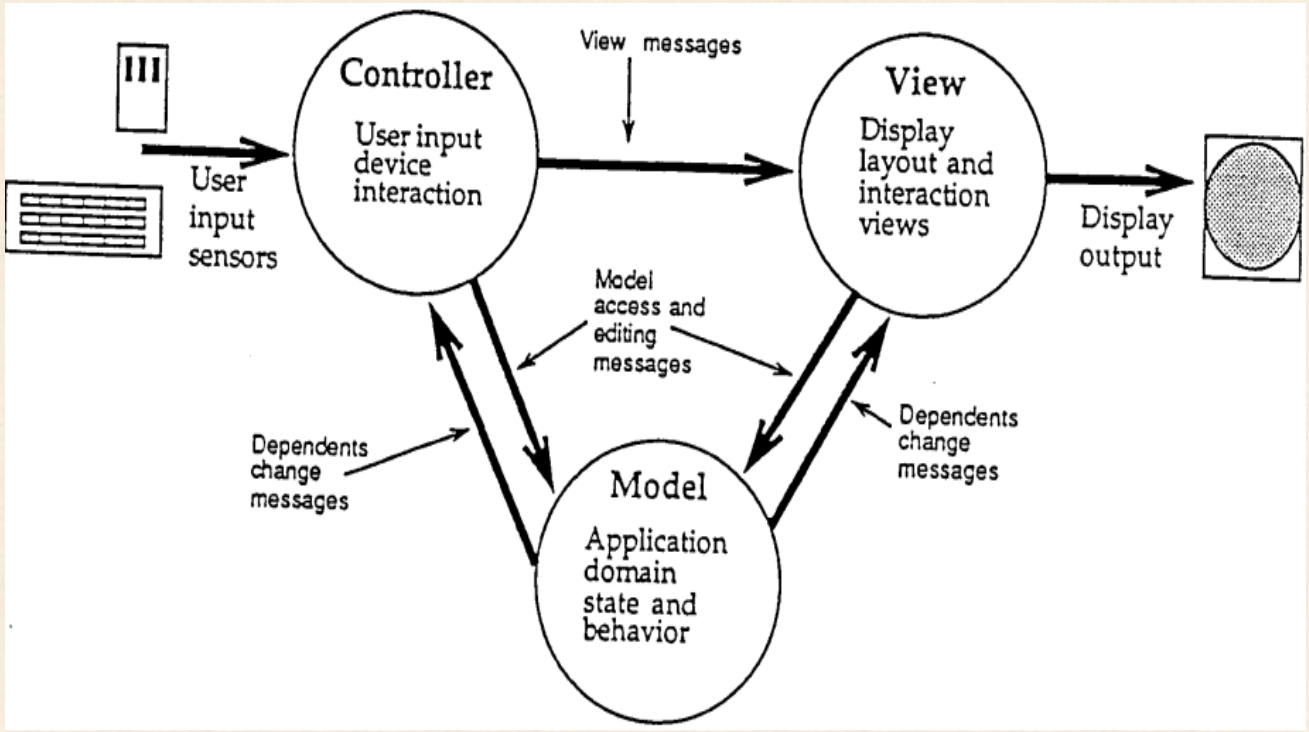
- ✓ Le bouton "***Chercher***" (Search Information) devient alors actif.
- ✓ À la réception du signal (émis par la Présentation) d'une pression sur le bouton "***Chercher***", le *Contrôle* demande à l'*Abstraction* la structure de requête et la fournit à son agent Père,
- ✓ puis demande à ses deux facettes *Abstraction* et *Présentation* de se réinitialiser.



2.2 Le modèle MVC:

Le modèle MVC montré dans la Figure ci-dessous propose une décomposition de l'application en trois parties. Cependant, dans ce modèle, le contrôleur n'a pas le même rôle que le contrôleur de PAC ou de Seeheim. Le Contrôleur de MVC contrôle les entrées des utilisateurs alors que la cohérence de l'application est assurée par la Vue (pour la partie graphique) et le Modèle (pour la partie sémantique).

MVC est le modèle multi-agents utilisé par Smalltalk et JAVA/SWING. Le but de ce modèle est d'avoir un système qui soit composé d'un ensemble de triplets autonomes et pouvant communiquer entre eux. Il est composé de trois éléments ou facettes qui sont :



- **Le Modèle (Model)** : qui est la structure de données que l'on veut représenter à l'écran et qui est composée d'objets, il maintient son état et notifie de ses modifications. (pour Smalltalk il représente les instances de classes)

- **La Vue (View) :** La vue écoute les notifications et interroge le modèle pour le représenter correctement, la Vue qui est la représentation externe du Modèle. C'est par elle que l'utilisateur perçoit le Modèle (inputs) et que le Modèle reflète ses changements (outputs). Grâce à la séparation des données et de leur représentation, un Modèle peut avoir plusieurs vues différentes alors qu'une Vue ne peut être associée qu'à un seul Modèle.

L'imbrication des Vues (donc le regroupement des Modèles) reste cependant possible, ce qui permet de développer des prototypes qui soient tous basés sur une même structure de données et de passer d'une représentation à l'autre sans changer le code de l'application sous-jacente. Les communications entre la Vue et le Contrôleur ne peuvent se faire sans passer par le Modèle qui est garant de la cohérence de l'état de l'agent.

- **Et le Contrôleur (Controller) :** Le Contrôleur écoute l'utilisateur et demande des modifications au Modèle, il est responsable de la régulation des interactions entre la Vue et le Modèle. Il est chargé de gérer les actions de l'utilisateur sur la Vue. Lorsque l'utilisateur manipule la Vue, le Contrôleur informe le Modèle des interactions faites. Ce dernier modifie alors son état et informe la Vue du nouvel aspect qu'elle doit prendre suite à cette modification.

  - ✓ Les limitations de ce modèle résident par exemple à la *gestion de la sélection*, i.e., le besoin d'une communication Vue/Contrôleur pour gérer le ***feedback***.., etc.,
  - ✓ Ces limitations sont résolues (dans les modèles postérieurs) par l'intégration de la *Vue* et le *Contrôleur* au sein d'une même facette (exemple : Java/SWING procède de cette manière).
  - ✓ MVC a été réutilisé dans le domaine des applications Web, mais avec modification vers MV.

3. Modèles hybrides

Les *modèles globaux* offrent une structuration globale de l'application mais pas de chacun des modules la constituant alors que les *modèles multi-agents* proposent une structuration de chaque agent mais pas de l'application dans son ensemble. Afin de tirer parti des avantages de ces deux types de modèles, les *modèles hybrides* ont été développés.

Les modèles hybrides sont des modèles qui globalement respectent un des modèles globaux et pour lesquels un (ou plusieurs) des modules issus de la décomposition globale est structuré, le plus souvent, en suivant le modèle d'architecture à agents.

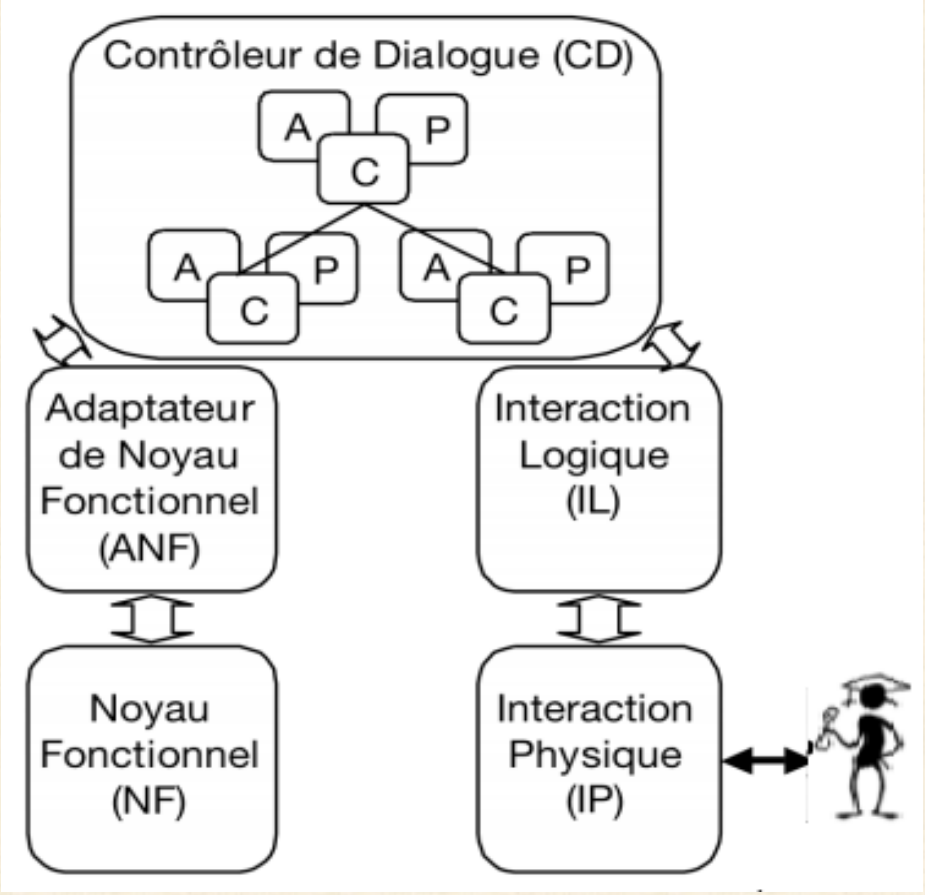
Parmi les modèles hybrides existants, on a **PAC-SEEHEIM** qui applique le modèle global **Seeheim** et le modèle multi-agents **PAC** sur chacun de ses modules, le modèle **H^4** qui représente une amélioration du modèle **ARCH** en s'aidant du modèle à agents **PAC**, un autre modèle d'architecture hybride, nommé **PAC-Amodeus** qui préconise la décomposition fonctionnelle d'**Arch** et intègre les capacités d'affinement et de prise en compte du parallélisme de **PAC**.

En ce qui suit nous allons détailler les deux derniers modèles.

3.1. PAC-Amodeus

PAC-Amodeus reprend les modules du modèle Arch dont il raffine le *Contrôleur de Dialogue* en *agents PAC*.
Les modules Le Noyau Fonctionnel (NF) et l'utilisateur sont les deux extrêmes du modèle.

- ✓ Dans ce modèle, le **NF** réalise les concepts du domaine en ignorant la façon dont ces concepts sont rendus perceptibles à l'utilisateur.
- ✓ Les autres modules du système, l'**Adaptateur** de **Noyau Fonctionnel** (ANF), le **Contrôleur de Dialogue** (CD) et les modules **d'Interaction Logique** (IL) et **Physique** (IP), modélisent l'IHM du système.



3.1. PAC-Amodeus (Cont.)

L’utilisateur et le NF produisent et consomment des informations par l’intermédiaire des modules de l'IHM. Ce fonctionnement symétrique laisse libre le choix du contrôle de l’interaction en fonction du cas à traiter : contrôle par l'utilisateur ou par le NF. Le NF échange des concepts du domaine (en fournissant des services) avec son voisin immédiat : l’adaptateur de noyau fonctionnel ANF.

L'**ANF** est une interface conçue pour absorber les changements entre ses voisins directs CD & NF.

Le modèle **PAC-Amodeus** utilise un ensemble de règles heuristiques adoptées par PAC (i.e., approuvées par des méthodes scientifiques) pour guider la structuration hiérarchique de son **CD** :

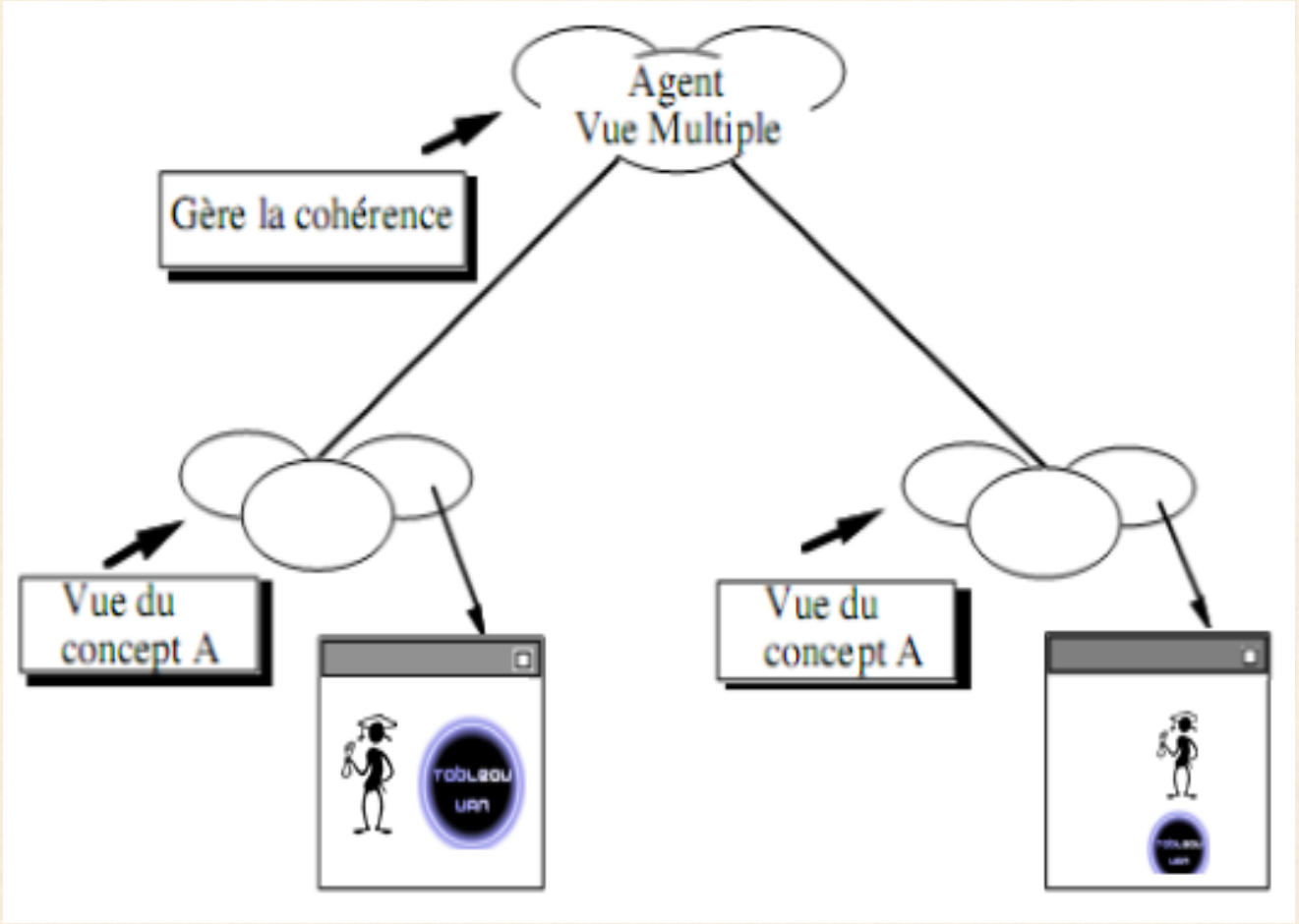
Règle 1 : Une fenêtre qui sert de support à un ‘espace de travail’ est modélisée par un agent.

3.1. PAC-Amodeus (Cont.)

Règle 2 : Les vues multiples d'un même concept (figure en bas) sont gérées par un agent "Vue Multiple" chargé de maintenir la cohérence entre les vues.

Exemple :

Un tableau de données peut être représenté graphiquement par: Un graphique à secteurs (agent1), à barres (agent2), à lignes (agent3), à colonnes (agent4)... etc.



3.1. PAC-Amodeus (Cont.)

Règle 3 : Une palette est modélisée par un agent.

Règle 4 : Une barre de menu est modélisée par un agent.

Règle 5 : Une zone d'édition est modélisée par un agent.

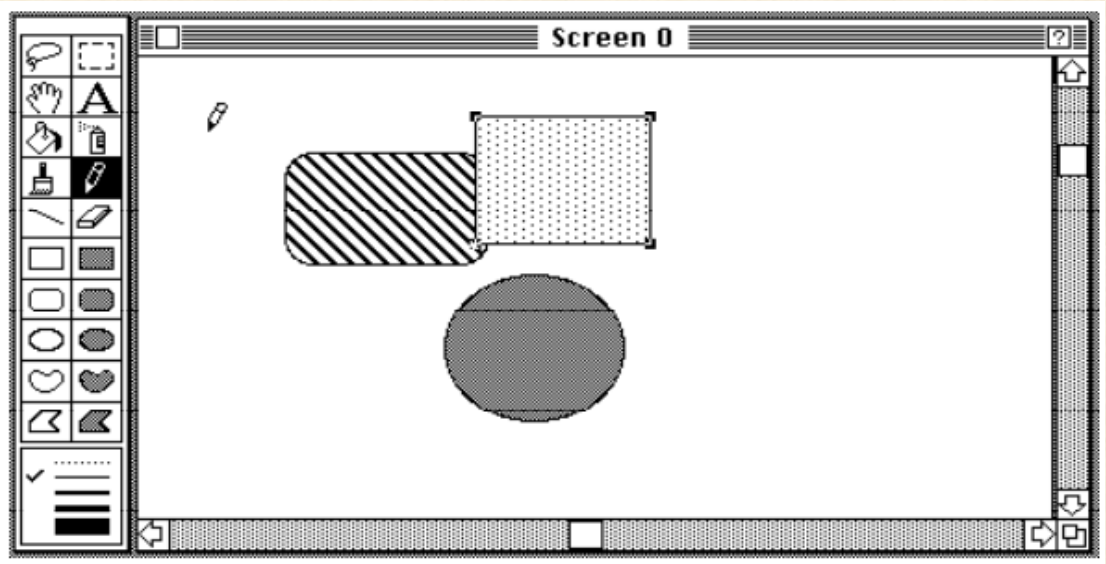
Règle 6 : Un concept complexe (ex., zone de dessin) est modélisé par un agent.

Règle 7 : Si une fenêtre "espace de travail" permet d'ouvrir une autre fenêtre "espace de travail" (i.e., une autre instance du même concept) les deux agents gérant ces deux fenêtres sont modélisés comme fils d'un même agent père.

Règle 8 : Si la nouvelle fenêtre représente moins ou plus de détails sur l'un des concepts, l'agent qui modélise cette fenêtre est fils de l'agent source.

3.1. PAC-Amodeus (Cont.)

Règle 9 : Si la spécification d'une commande implique des actions distribuées sur plusieurs agents, ceux-ci doivent être placés sous le contrôle d'un agent qui cimente (figure en bas) les actions réparties en une commande.



Règle 10 : Un agent PAC et son fils unique peuvent être regroupés en un seul agent.

Règle 11 : Un agent dont le rôle peut être encapsulé par un objet de présentation ou d'interaction peut être éliminé et apparaître comme un composant dans la présentation de son agent père.

3.2. Le modèle H^4

Le modèle hybride **H^4** (Figure ci-dessous) est basé sur le modèle global **ARCH** (bien que les noms des modules différent sensiblement).

- ✓ **H^4** propose une structure hiérarchique en agents pour *quatre des cinq* modules de l'**Arch** : le *Noyau Fonctionnel*, le *Contrôleur de Dialogue*, *l'Adaptateur de Présentation* (IL de l'Arch) et la *Présentation* (IP de L'Arch).
- ✓ Ces quatre modules sont structurés de manière hiérarchique telle que chaque niveau de la hiérarchie étant une spécification du niveau d'abstraction supérieur.