

# Langage formel et heuristiques pour des agents conversationnels assistants

**François Bouchet**

[bouchet@limsi.fr](mailto:bouchet@limsi.fr)

Directeur de thèse : Jean-Paul Sansonnet

Ecole Doctorale d'Informatique de Paris-Sud – Bourse MENRT

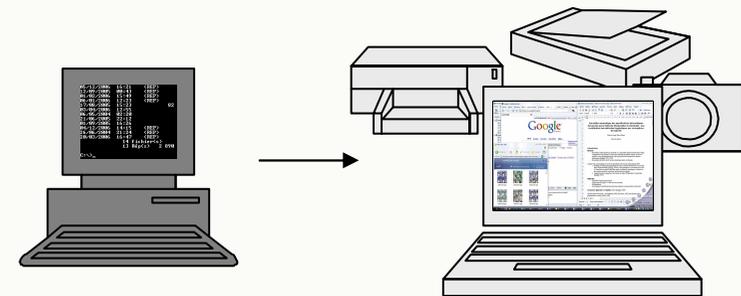
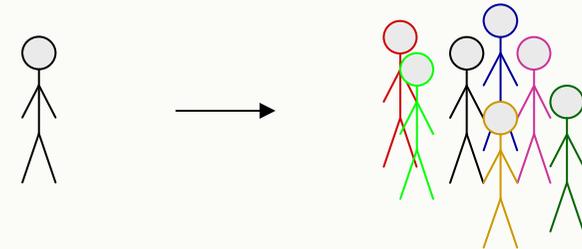
**Journée des Doctorants 2007**

12 janvier 2007



# Contexte : le besoin d'assistance

- La population d'**utilisateurs** d'outils informatiques a considérablement évolué en 30 ans :
  - ◆ En **nombre** : la moitié des ménages français ont au moins un ordinateur [INSEE, 2005].
  - ◆ En **diversité** : informatique d'informaticiens → informatique pour tous.
- Les **machines** sont plus puissantes (loi de Moore) et ont élargi ainsi :
  - ◆ Leur **champ d'application** : traitement de texte, multimédia, jeux vidéos, recherche d'information...
  - ◆ Leur **champ d'interaction** : imprimante, scanner, PDA, appareil photo...
- Les **logiciels** sont devenus :
  - ◆ Plus **nombreux** : dizaines de traitements de texte ou d'applications d'édition de photos.
  - ◆ Plus **complexes** : 45 boutons et 150 actions « de base » accessibles par 9 menus dans Word 2003.



Inadéquation croissante entre le système et la représentation cognitive qu'en ont les utilisateurs.

# Le projet Daft

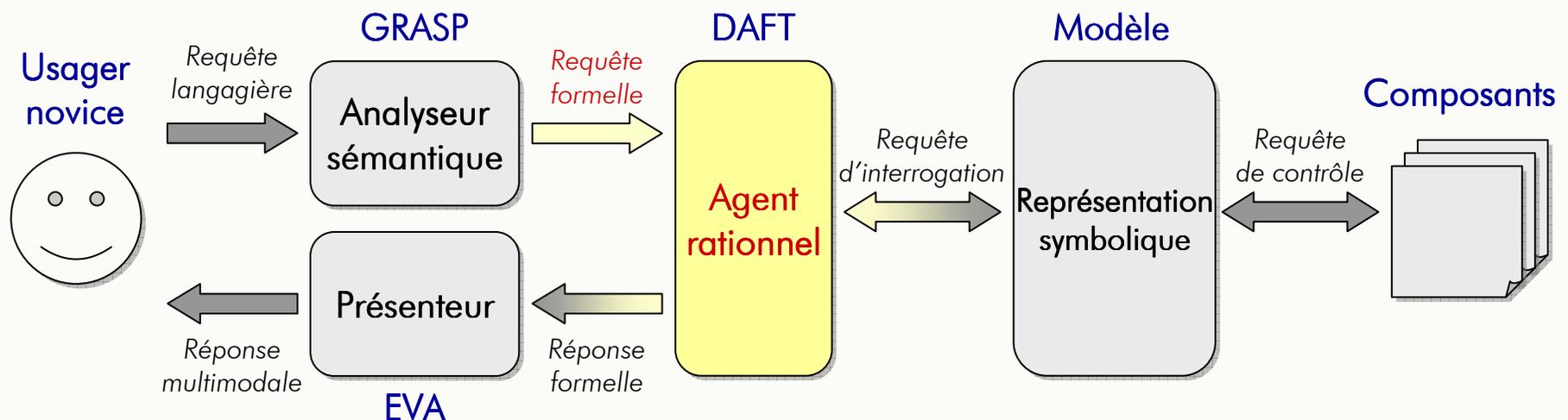
## ■ Objectifs :



### ◆ Agents Conversationnels Animés dédiés à la fonction d'assistance :

- Pour : les utilisateurs novices s'exprimant en langue naturelle non contrainte
- Par : un raisonnement sur la structure et le fonctionnement des applications (aide contextuelle)

## ■ Schéma d'architecture générale :



# Problématique de la thèse

- Bilan du stage de M2R : un langage de requêtes formelles :
  - ◆ Ancré dans la réalité : étude d'un corpus de 8000 requêtes.
  - ◆ Permettant une expressivité fine de la sémantique des requêtes issues des usagers.
- Objectifs de la thèse :
  - ◆ **Implémentation** du langage de requêtes formelles.
  - ◆ Interprétation pragmatique des requêtes à l'aide d'**heuristiques**.
  - ◆ Un système flexible pour expérimenter la pertinence des heuristiques auprès d'utilisateurs réels.

# Des heuristiques au cœur de l'agent assistant

**Heuristiques** = règles symboliques canoniques (ne dépendant ni des usagers ni des composants qu'elles assistent).

1. **Lire** une requête formelle ;
2. **Contextualiser** la requête :  
→ Exploration des modèles ;
3. **Traiter** la requête :  
→ Mise à jour des modèles ;
4. **Produire** la réponse formelle.

« y a combien de boutons? »

**ASK [ COUNT [ REF ( BUTTON ) ] ]**

**Langage formel de requêtes**



**Agent assistant**  
 $\Sigma H_i$

L'agent introspecte les modèles, raisonne dessus et peut même les modifier pour résoudre la requête



**Langage formel de réponses**

**TELL [ EQUAL [ COUNT [ REF ( BUTTON ) ] ] , 3 ] ]**

« Il y a trois boutons »



L'utilisateur produit un modèle d'interaction qui lui est spécifique

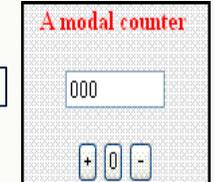
**Modèle d'interaction**

- Session dialogique
- Profil usager
- Tâche en cours



**Modèle du composant**

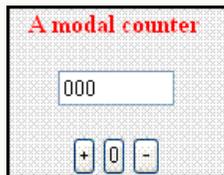
- Structure
- Actions
- Processus



Le composant produit un modèle qui lui est spécifique

# Classes d'équivalence

- Principe : N formulations  $\Leftrightarrow$  1 unique sens en contexte
- Différents niveaux :
  - ◆ D'ordre **grammatical** :
    - « la vitesse est changée par le curseur »  $\Leftrightarrow$  « le curseur change la vitesse »
  - ◆ D'ordre **sémantique** :
    - **ASK**(**POSSIBILITY**(todo=...))  $\Leftrightarrow$  **WILL**(todo=...)
  - ◆ D'ordre **pragmatique** : utilisation de règles de sens commun (Grice)



- « active le compteur »
- « démarre l'appli » (*appli = compteur*)
- « vas-y » (*action par défaut = lancer le compteur*)
- « dommage que le compteur soit bloqué »

$\Leftrightarrow$  **WILL**(of=user,  
todo=**Start**(object="compteur"))

# Heuristiques de réactions pragmatiques

## ■ Définition de réactions pour des classes :

- ◆  $REQUEST_{user} [A(x,y)] \rightarrow INTENT_{syst} [ EXECUTE[A(x,y)] ]$
- ◆  $JUDGE_{user} [*J(x)] \rightarrow INTENT_{syst} [ rev(J,x) (x) ]$

*\*: péjoratif*

**P: proposition**

**A: action**

**J: jugement**

**rev: opérateur de réversion**

## ■ Exemples :

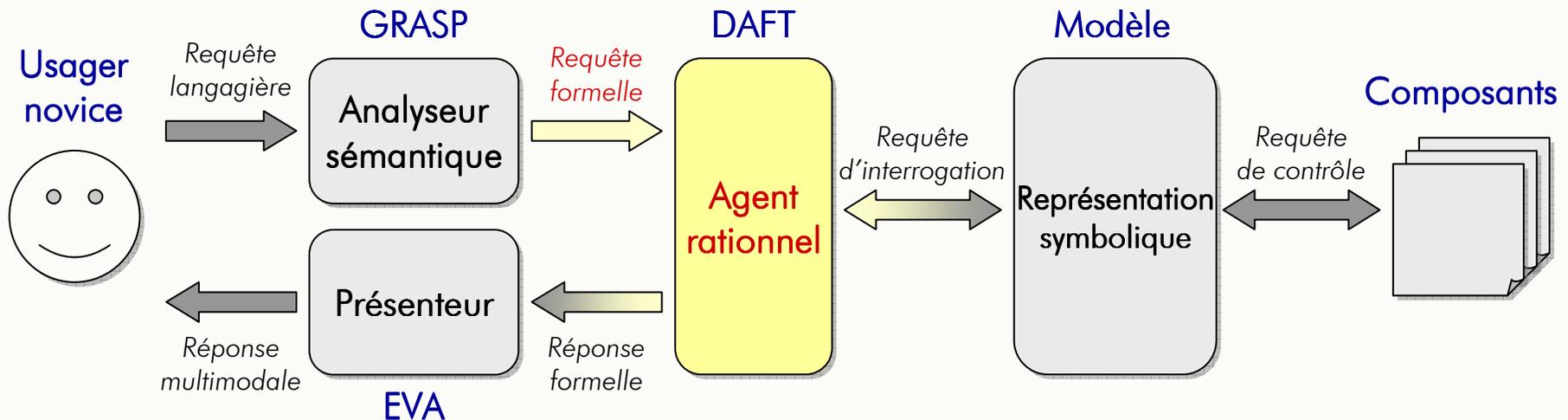
- ◆ « tu parles trop fort »
  - $JUDGE_{user} [*TOOMUCH(system.sound.level)]$
  - $rev(TOOMUCH, x:NUM) = EXECUTE[DECREASE(x, 20\%)]$

Diminution du volume

- ◆ « j'aime pas tes yeux »
  - $JUDGE_{user} [*LOOK(system.bodypart.eye)]$
  - $INTENT_{syst} [ rev() (x) ]$
  - $ANSWER_{syst} [< No modifying operator for 'LOOK' >]$

« Désolé, je ne puis modifier mon apparence »

# Conclusion



- Travail réalisé :
  - ◆ Etude du corpus de requêtes utilisateurs.
  - ◆ Spécification d'un langage de requêtes formelles lié aux spécificités du corpus.
- Travail en cours :
  - ◆ **Implémentation** du langage de requêtes formelles.
- Cœur de la thèse :
  - ◆ Interprétation pragmatique des requêtes à l'aide d'**heuristiques**.
  - ◆ Contrainte : système flexible pour expérimentations.