Apprentissage par renforcement de modèles de contexte pour l'informatique ambiante

Sofia ZAIDENBERG
Laboratoire d'Informatique de Grenoble
Équipe PRIMA

Jury composé de

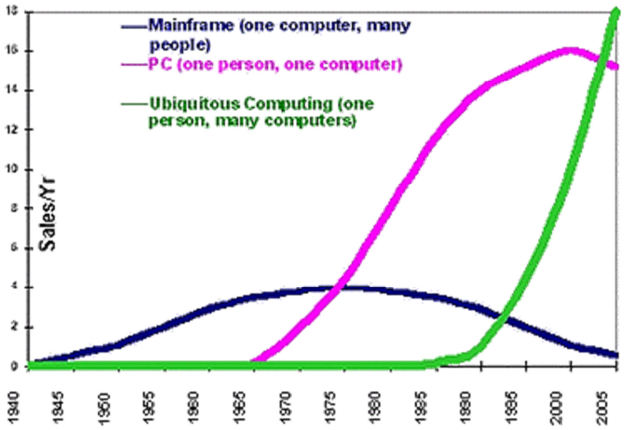
Mme Brigitte PLATEAU	Présidente du jury
M. Olivier SIGAUD	Rapporteur
M. Olivier BOISSIER	Rapporteur
M. James L. CROWLEY	Directeur de thèse
M. Patrick REIGNIER	Co-directeur de thèse
Mme Marie-Pierre GLEIZES	Examinatrice

16 octobre 2009

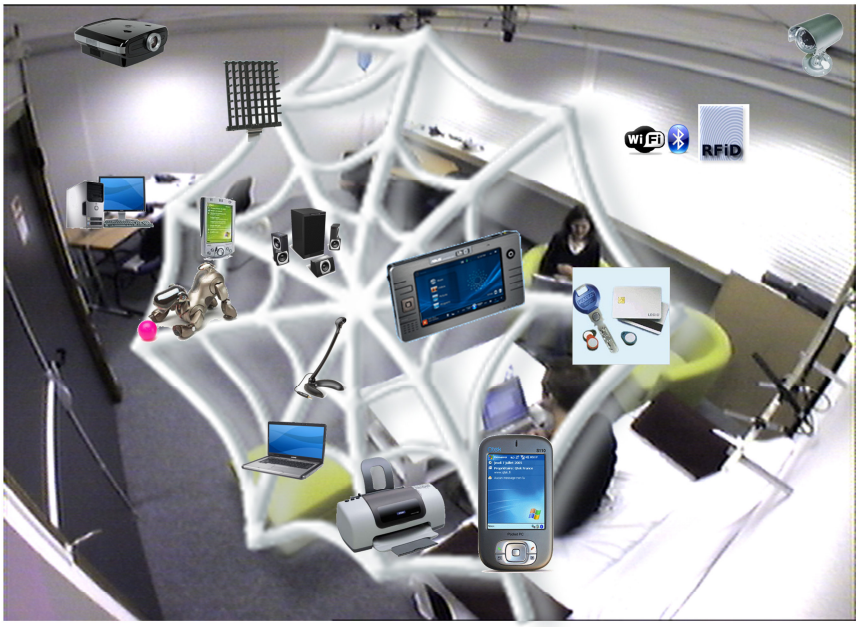
Informatique ambiante

Informatique ubiquitaire

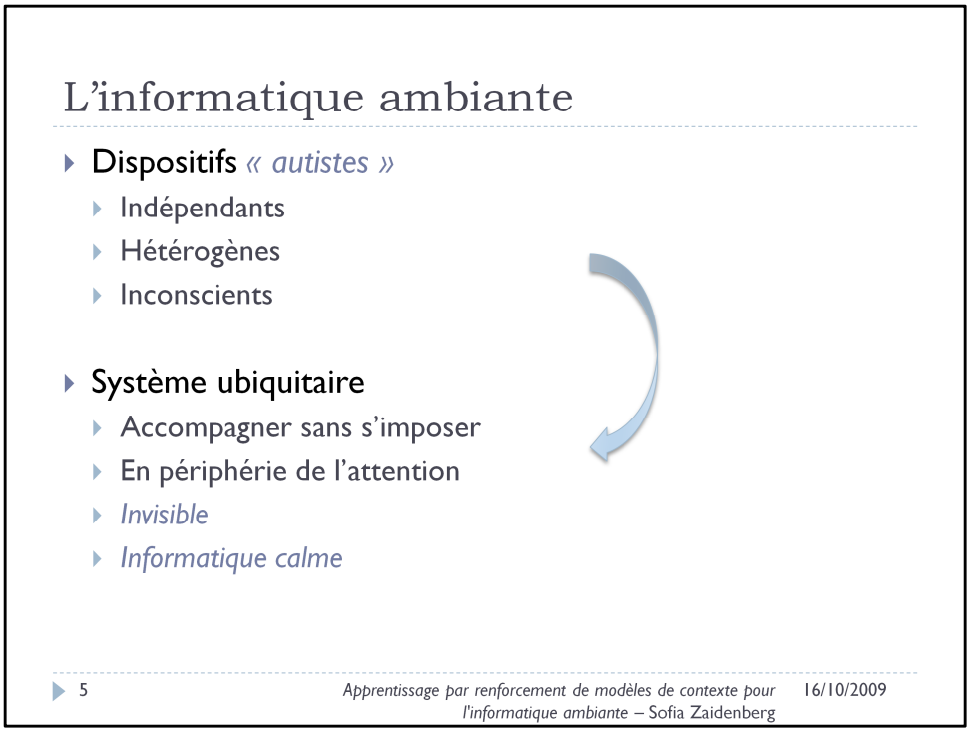
[Weiser, 1991]
[Weiser, 1994]
[Weiser et Brown, 1996]



Environnements quotidiens sont déjà remplis de dispositifs, qui ont des moyens de se connecter les uns aux autres.



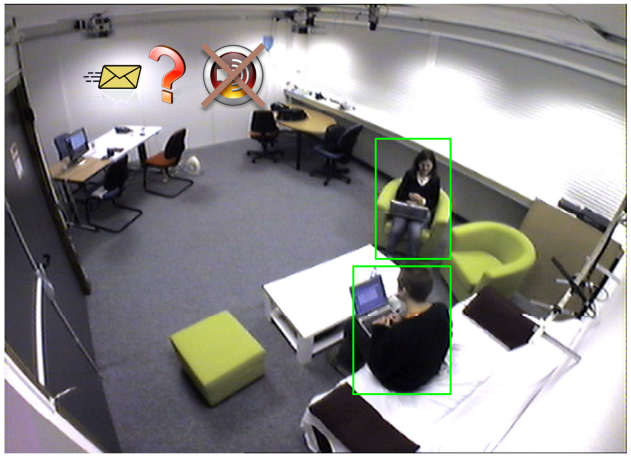
Ces dispositifs sont isolés, ils ont été rapportés par différentes personnes, à différents moments…
On voudrait qu'ils se rapprochent les uns des autres, et aussi des autres dispositifs dans le bâtiment pour former un ensemble cohérent, pour les faire coopérer, les synchroniser, leur faire former un ordinateur virtuel, une intelligence commune dispersée (distribuée) dans tous ces dispositifs atomiques.



Volonté de les faire coopérer les dispositifs pour créer un ensemble cohérent, non-intrusif, pour rendre un service unifié à l'utilisateur.

Intelligence ambiante

- **Context-aware computing**



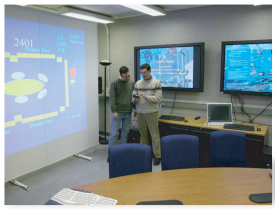
1. Perception
2. Décision

Intelligence = Percevoir l'environnement et la situation
Sélectionner et exécuter une action adéquate qui
Modifie l'environnement
ou bien
Modifie le système lui-même
Afin de se trouver dans un état désirable (par rapport à un certain but ou critère).

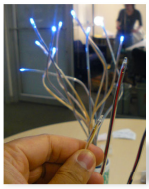
Intelligence dans un système ubiquitaire :
Percevoir la situation de l'utilisateur (le **contexte**)
Lui rendre un service adéquat

État de l'art

[Nonogaki et Ueda, 1991]
FRIEND21



[Roman *et al.*, 2002]
Gaia



Blossom

Sajid SADI et Pattie MAES

http://consciousanima.net/projects/blossom/

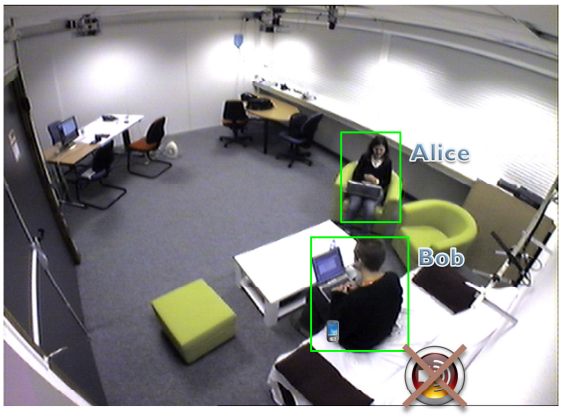
FRIEND21 : recherche sur les principes à respecter pour réaliser les interfaces du 21ème siècle. Communication et accès à l'information facilités et prise en compte du contexte.

Gaia : un système d'exploitation gérant un « espace actif », en prenant en compte le contexte. Gère les incertitudes de la perception par apprentissage mais pas d'apprentissage des préférences. Interaction transparente étendue sur tous les dispositifs de l'environnement.

Blossom (MediaLab du MIT) : permet de garder le contact avec ses proches de manière non-intrusive, en se focalisant sur la présence en périphérie de l'attention plutôt que sur une communication directe.

Problématique

- Personnalisation
  - Situation + utilisateur ⇨ action



But : obtenir un système ubiquitaire aussi personnalisé qu'un ordinateur (couleurs, fontes, icônes, etc.)

Exemple : Toujours mettre le téléphone de Bob en vibreur lorsque d'autres personnes sont présentes, toujours en sonnerie lorsqu'il est seul.

Personnalisation

- Personnalisation d'un agent informatique complexe qui assiste l'utilisateur.
- Deux solutions [Maes, 1994]
  - L'utilisateur spécifie lui-même le comportement
    - Système trop complexe ⇨ Tâche laborieuse
    - Peu-évolutif
  - Choix prédéfini par un expert
    - Non-personnalisé
    - Non-évolutif
    - Utilisateur ne maîtrise pas tout le système

Solution proposée

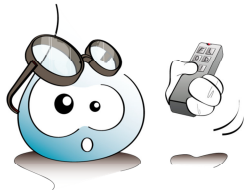
Apprentissage

Plan

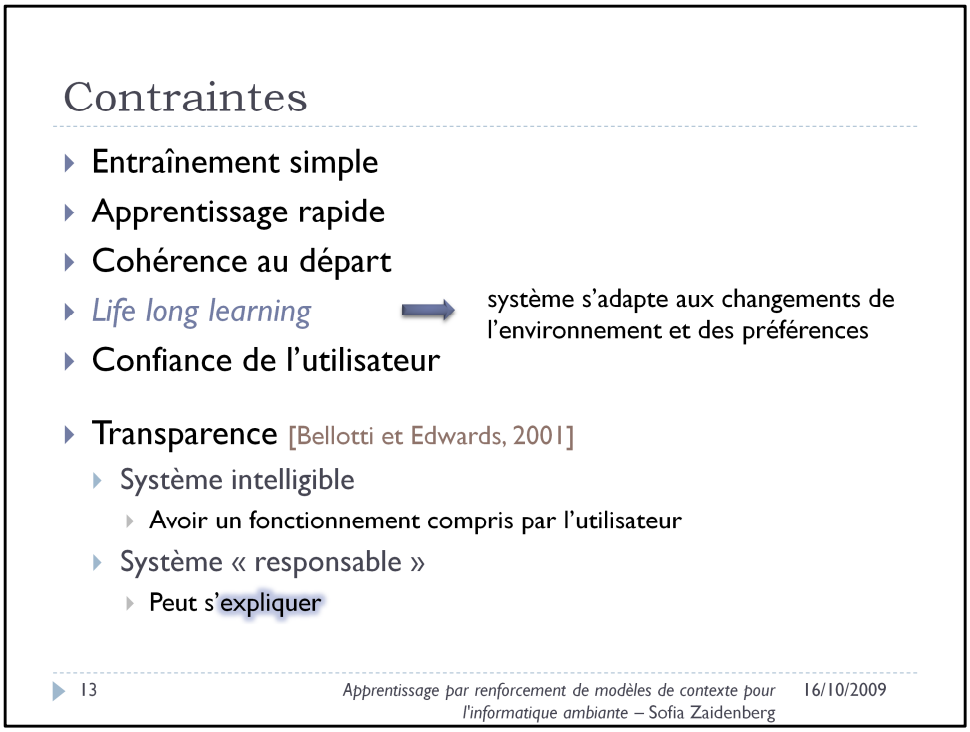
- Présentation du problème
- **Apprentissage dans les systèmes ubiquitaires**
- Enquête grand public
- Réalisation d'un système ubiquitaire
- Apprentissage par renforcement du modèle de contexte
- Expérimentations et résultats
- Conclusion

Système proposé

- Un **assistant virtuel** qui personnifie le système ubiquitaire
- L'assistant
  - Perçoit le contexte grâce aux capteurs
  - Exécute des actions grâce aux actionneurs
  - Reçoit les retours de l'utilisateur pour l'entraînement
  - Adapte son comportement à ces retours *(apprentissage)*



Apprendre les préférences individuelles au fur et à mesure des interactions de l'utilisateur avec l'environnement.



Entrainement non-intrusif, peu contraignant.

Confiance : comment faire en sorte que l'utilisateur se sente à l'aise avec un système qui fait des choses pour lui ?
Risque si on n'a pas gagné sa confiance : rejet du système.

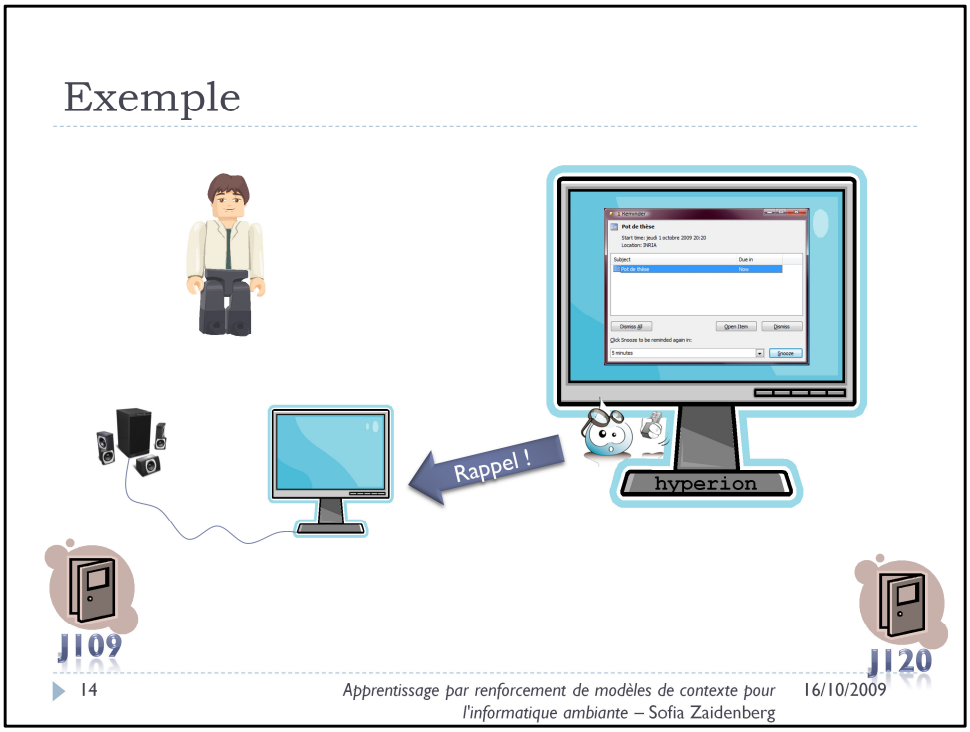
Apprentissage permet une personnalisation avec un moindre effort de la part de l'utilisateur.

Transparence (le système ne doit pas être une boîte noire) :
Système qui agit en fonction du contexte ne peut pas prétendre percevoir le contexte de manière parfaite, donc il doit prendre ses responsabilités par rapport à ça :
être intelligible + responsable : savoir expliquer à l'utilisateur

Quelle est sa connaissance
Comment l'a-t-il obtenue
Quelles sont ses actions, et pourquoi les a-t-il choisies

Système évolue pas à pas

⇨ l'utilisateur a le temps de s'habituer (permet en partie de gagner sa confiance).



L'assistant personnel s'exécute sur la machine de bureau de l'utilisateur. Il détecte un rappel de l'agenda et décide de le transmettre à l'utilisateur. L'utilisateur se trouve dans le bureau J109, qui est équipé de haut-parleurs, l'assistant connait la machine à laquelle ils sont reliés, il peut donc lui envoyer le texte qui sera prononcé par synthèse vocale. L'utilisateur peut ensuite donner son avis sur ce service.

Plan

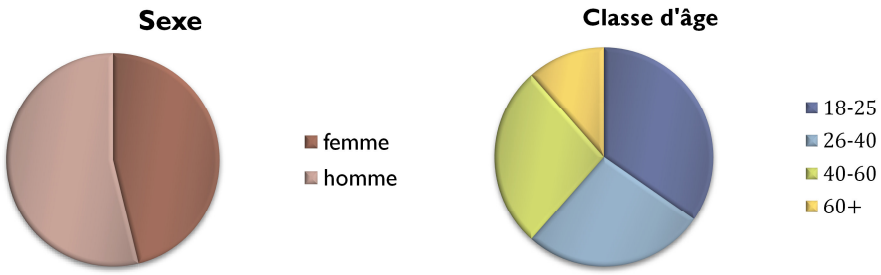
- Présentation du problème
- Apprentissage dans les systèmes ubiquitaires
- **Enquête grand public**
- Réalisation d'un système ubiquitaire
- Apprentissage par renforcement du modèle de contexte
- Expérimentations et résultats
- Conclusion

Enquête grand public

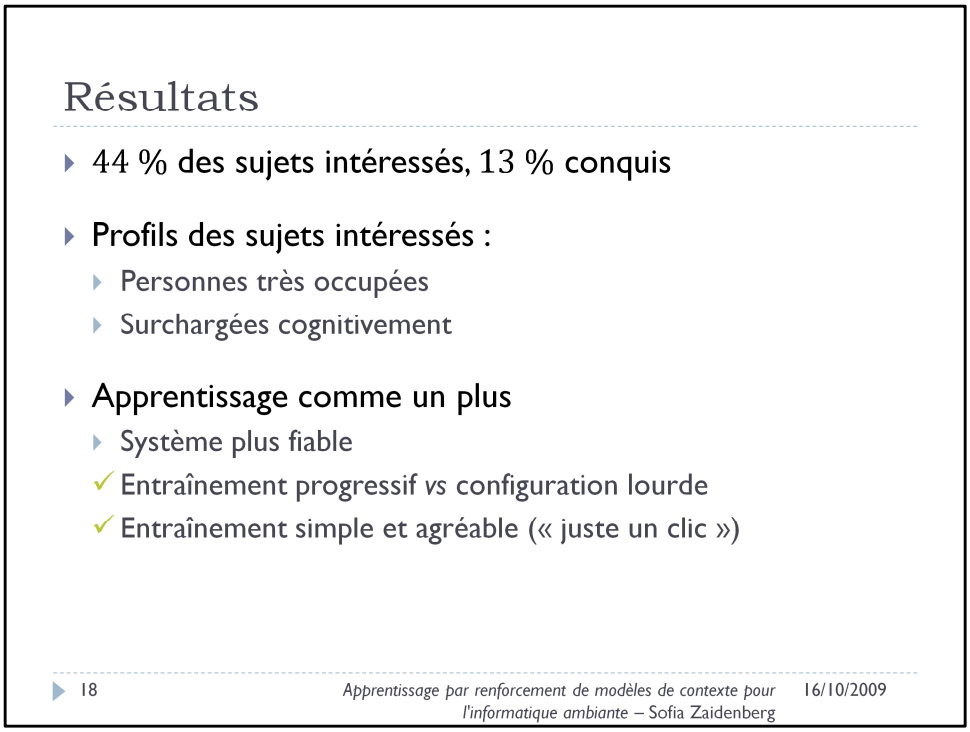
- Objectif
  - Mesurer les attentes et besoins vis-à-vis de l'« informatique ambiante » et de ses usages
- Enquête dirigée par Nadine Mandran (LIG)
- Évaluation simultanée de deux systèmes
  - Notre assistant
  - Système COMPOSE de Yoann Gabillon (MAGMA et IIHM)
    - Composition (semi-)automatique, dynamique et contextuelle de services pour répondre aux requêtes utilisateur

Modalités de l'enquête

- 26 sujets interrogés
  - Non-experts
  - Répartis de manière suivante :



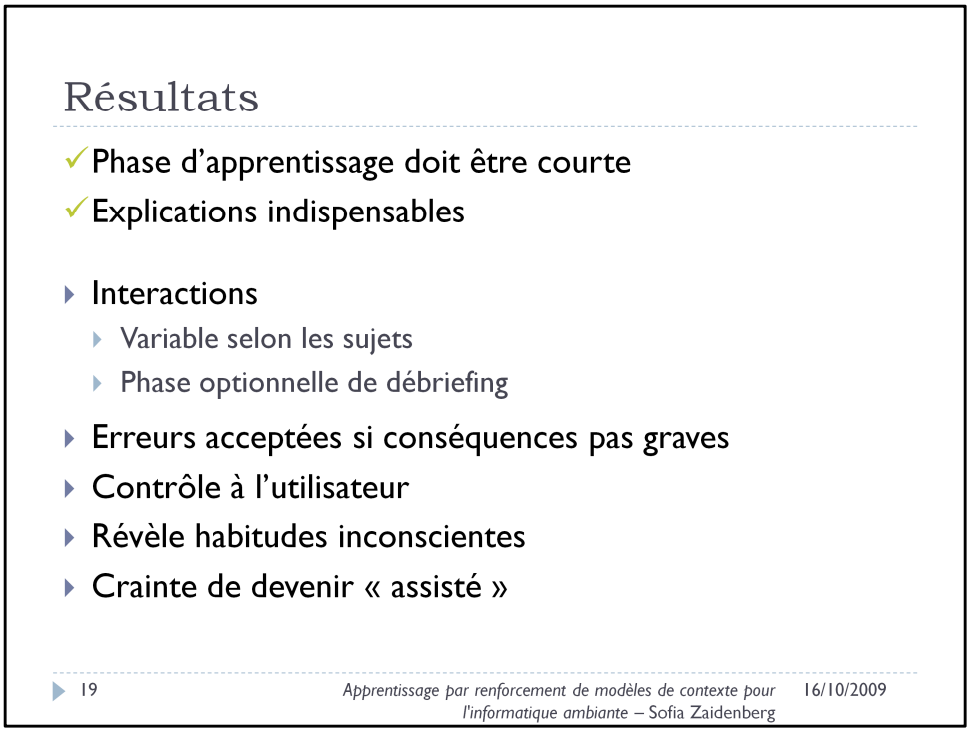
Surjets non-informaticiens.



Profils des sujets intéressés : personnes ayant un emploi du temps très chargé et dynamique, mêlant vie personnelle et professionnelle, personnes souhaitant une aide à l'organisation et à la gestion du temps.

Erreurs acceptées si l'utilisateur sait que le système apprend et si le système apporte un plus.

Cette enquête permet de justifier notre recherche, donc on a cherché à savoir si nos contraintes étaient bonnes et s'il y a d'autres éléments à prendre en compte.



La phase d'apprentissage (initiale) doit être courte (une à trois semaines).

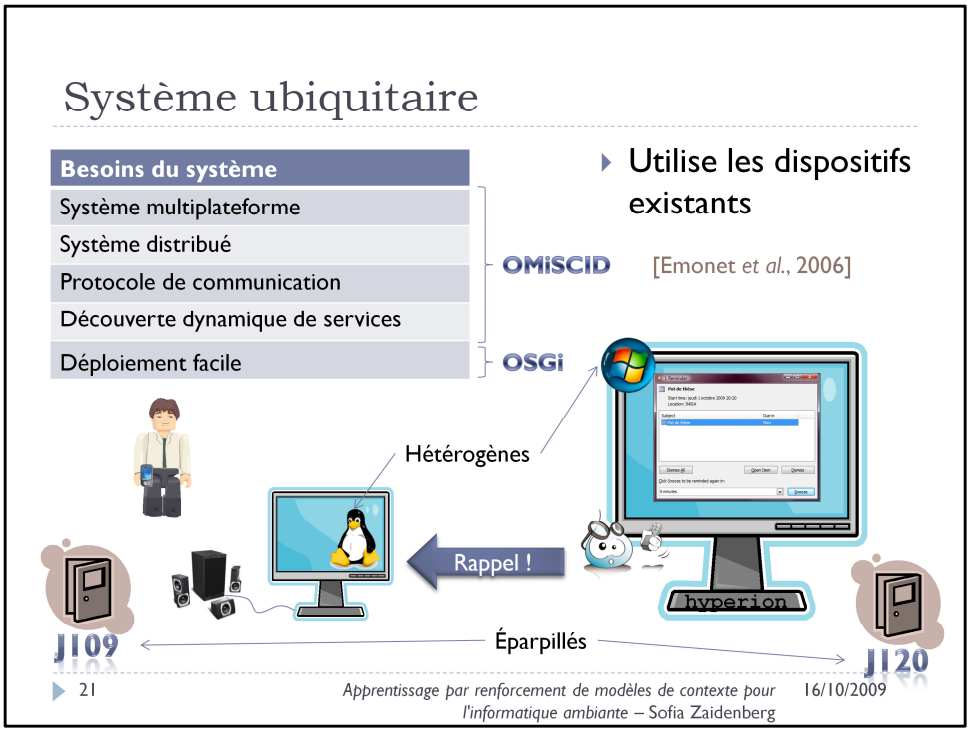
L'utilisateur doit toujours avoir le dernier mot, il doit garder le contrôle, il doit pouvoir éteindre tout le système par un geste (par exemple un « bouton rouge »), immédiatement.

L'assistant permet à l'utilisateur de découvrir ses propres modes de fonctionnement automatique dont il n'a pas toujours conscience.

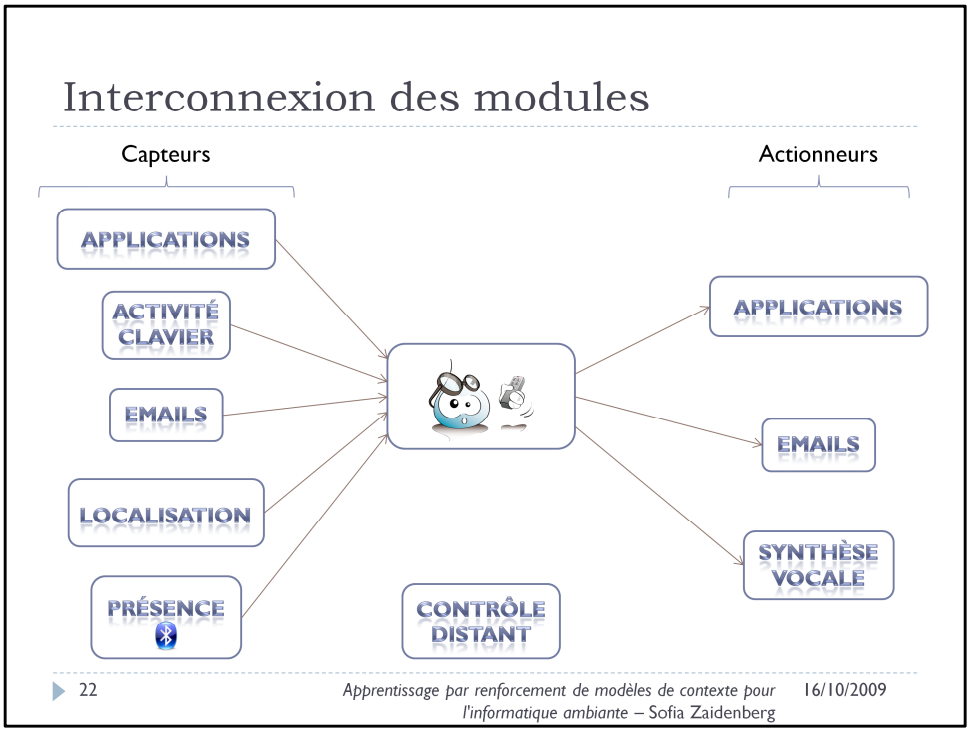
Peur de devenir dépendant d'un système qui fait des choses à notre place (quoi faire en cas de panne du système ?).

Plan

- Présentation du problème
- Apprentissage dans les systèmes ubiquitaires
- Enquête grand public
- **Réalisation d'un système ubiquitaire**
  - Contraintes
  - Technologies adoptées
- Apprentissage par renforcement du modèle de contexte
- Expérimentations et résultats
- Conclusion



Besoin de déploiement facile car on veut un système qui marche ne permanence (système vivant en permanence), on veut jamais l'arrêter pour maintenance.
→ Déploiement à chaud
→ Administration à distance
→ Gestion des modules à partir d'un dépôt central → pratique pour mettre à jour et pour ajouter dynamiquement de nouvelles fonctionnalités sans intervention manuelle.



Actionneur applications : permet de piloter des programmes en utilisant des systèmes standard d'exportation de fonctionnalités offerts par les systèmes (ex : dcop dans KDE).

Un module du système ambiant est un bundle OSGi et un service OMISCID à la fois.

Chaque dispositif appartenant au système a une plateforme OSGi (Oscar) dans laquelle au moins 2 modules sont installés et démarrés : remoteShell et OMISCID, et (stratégie opportuniste), on peut dynamiquement et automatiquement installer d'autres selon les besoins → permet d'obtenir un environnement souple, fluide.

Le module remoteShell sur chaque plateforme permet de contrôler le cycle de vie des autres bundles de la plateforme (installation, démarrage, mise-à-jour).

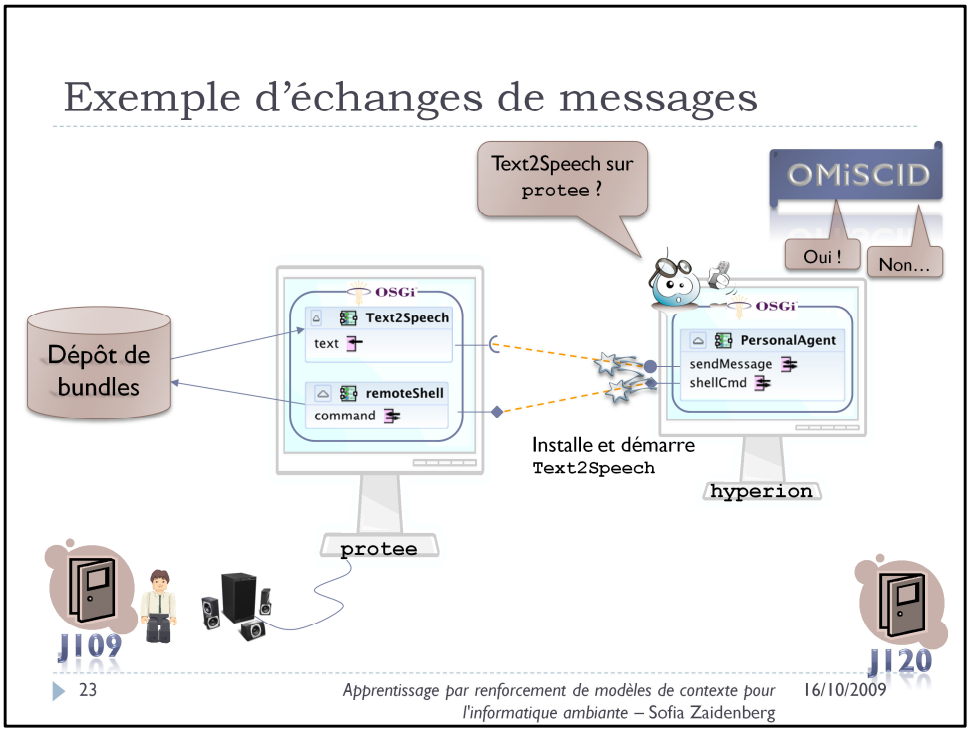
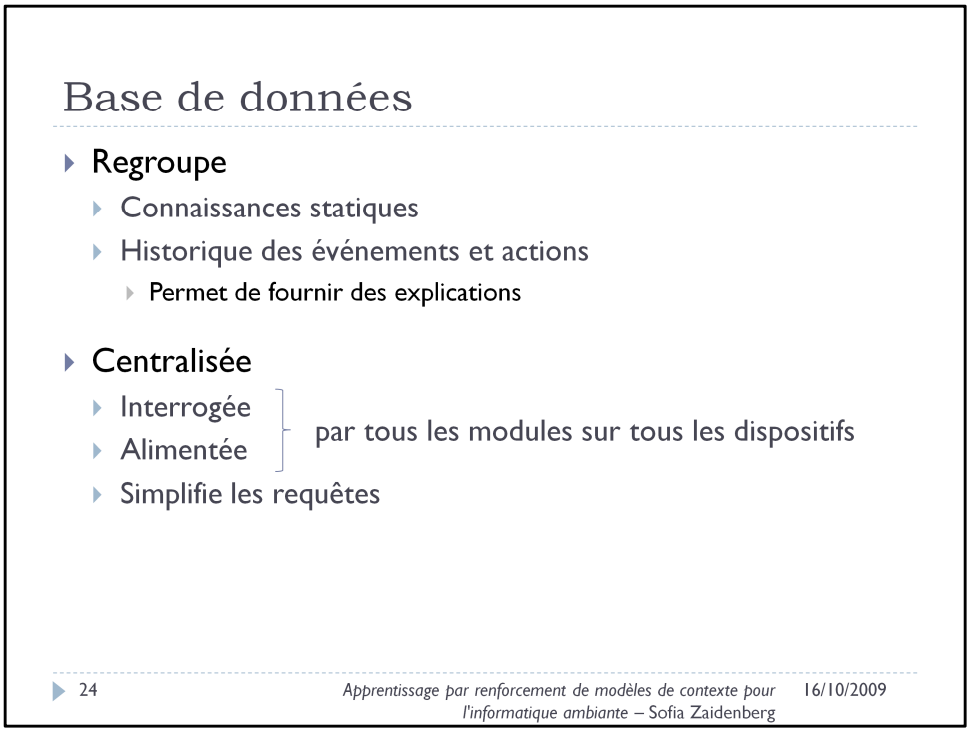


Illustration du déploiement à chaud (stratégie opportuniste) :
Exemple de transmission du rappel vu à un niveau plus bas (implémentation).

Le module AssistantPersonnel (« PersonalAgent ») s'exécute sur la machine de bureau de l'utilisateur. Il veut contacter le module « Text2Speech » (module de synthèse vocale) sur la machine « protee » (celle connectée aux haut-parleurs du bureau dans lequel se trouve l'utilisateur). C'est OMiSCID qui permet de brancher les modules les uns aux autres. Dans un 1er temps, OMiSCID répond à l'assistant que Text2Speech n'est pas en exécution sur protee. Alors l'assistant demande à OMiSCID d'obtenir une référence vers remoteShell (administration à distance) sur protee. On suppose que remoteShell doit toujours être disponible. L'assistant se branche sur remoteShell et lui envoie une commande pour installer et démarrer Text2Speech. remoteShell le fait à partir du dépôt central. Maintenant l'assistant redemande à OMiSCID une référence vers Text2Speech sur protee et l'obtient. Il s'y branche et lui envoie le texte à prononcer.

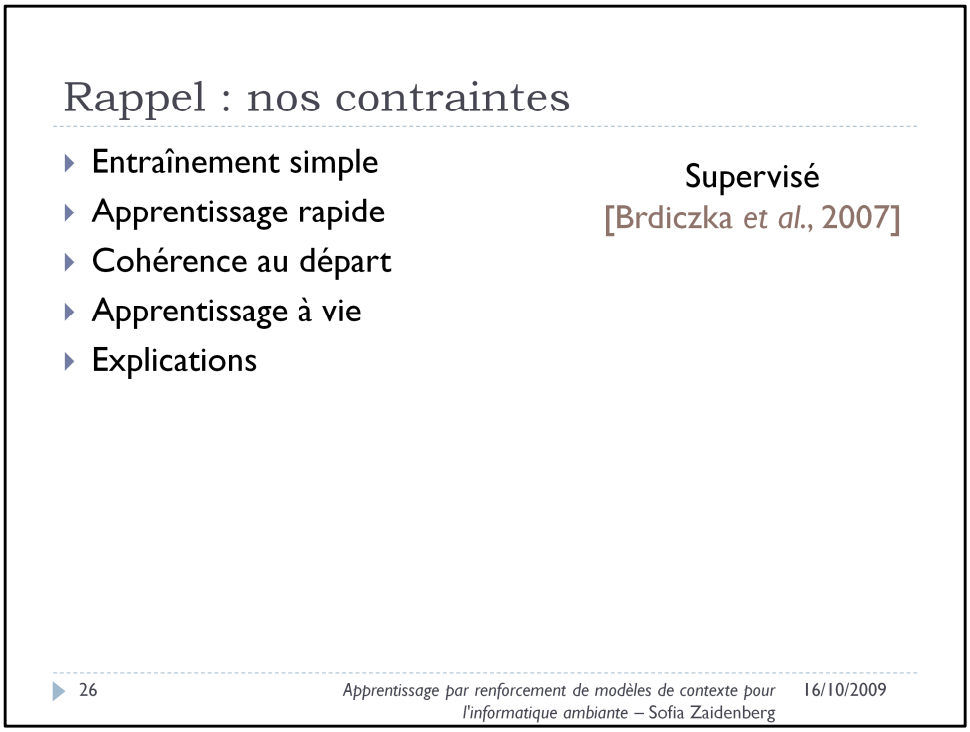


Connaissances statiques sur l'infrastructure et les utilisateurs (bureaux, adresses mail, dispositifs Bluetooth, etc.)

Historique des événements et actions du système et du cycle de vie des modules.

Plan

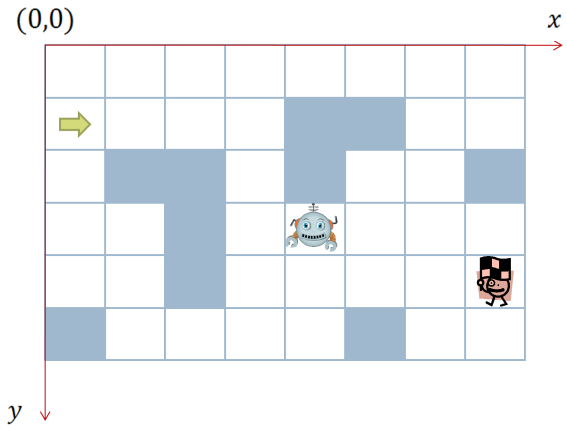
- Présentation du problème
- Apprentissage dans les systèmes ubiquitaires
- Enquête grand public
- Réalisation d'un système ubiquitaire
- **Apprentissage par renforcement du modèle de contexte**
  - Apprentissage par renforcement
  - Application de l'apprentissage par renforcement
- Expérimentations et résultats
- Conclusion



Apprentissage supervisé oblige l'utilisateur à étiqueter les séquences passées après coup. On préfère apprendre au fur et à mesure, que l'utilisateur puisse donner son retour « à chaud ».

L'apprentissage par renforcement est adapté (ou peut l'être avec les modifications que nous allons apporter) à ces contraintes.

Apprentissage par renforcement (AR)



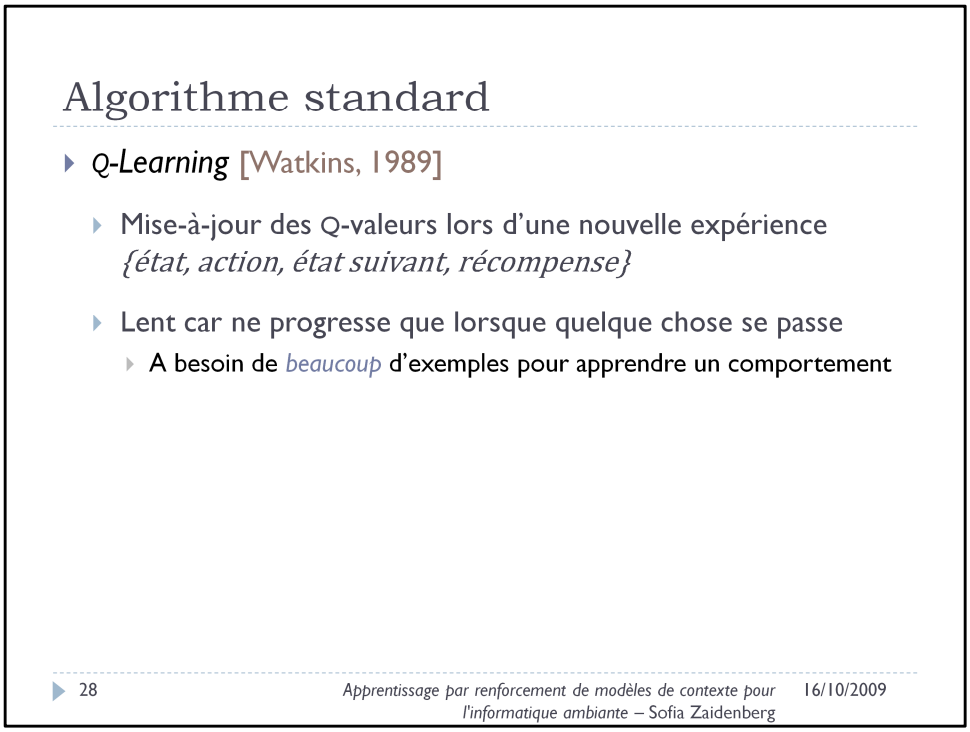
$Q(état, action)$

- Propriété de Markov
  - L'état à l'instant t ne dépend que de l'état à l'instant t-1

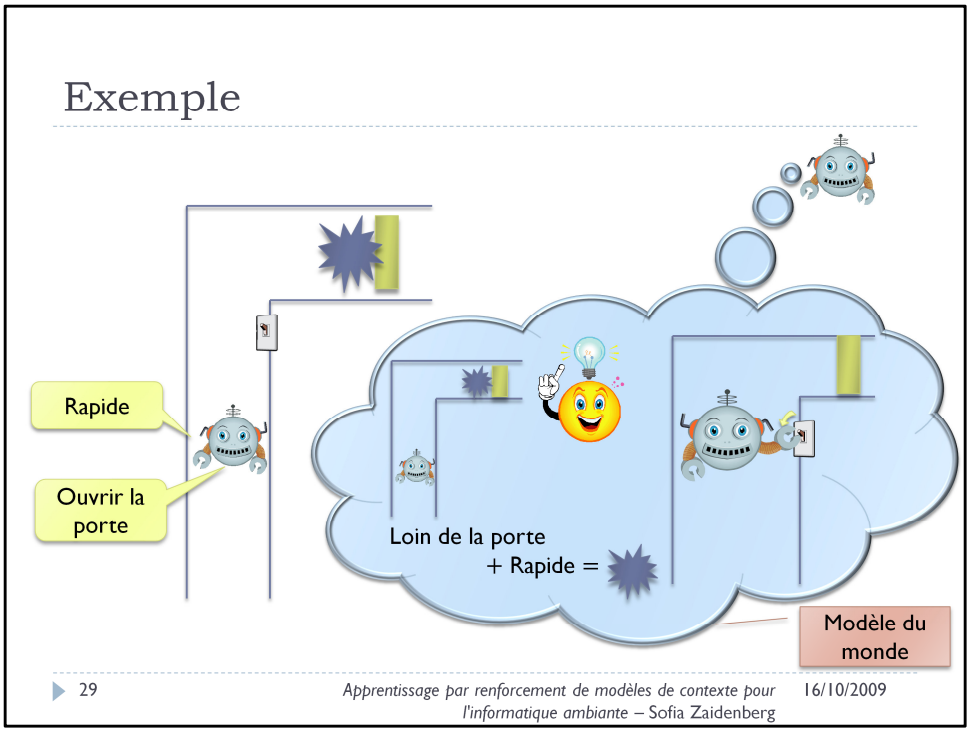
L'environnement est le labyrinthe, le robot connait son état dans l'environnement : (x, y), et peut faire des actions (se déplacer d'une case), reçoit des renforcements (but = très bien, dans le mur = très mauvais), doit maximiser la somme des renforcements reçus dans le temps.

Il explore l'environnement pour apprendre toutes les valeurs de qualité des couples état-action.

L'AR (apprentissage par renforcement) est bien adapté à des environnements contraintes, entièrement connus et maitrisés (potentiellement stochastiques). La difficulté de ce travail consiste à l'adapter à toutes les contraintes introduites par un environnement réel, complexe, et le fait que l'utilisateur est directement impliqué dans les actions de l'assistant.



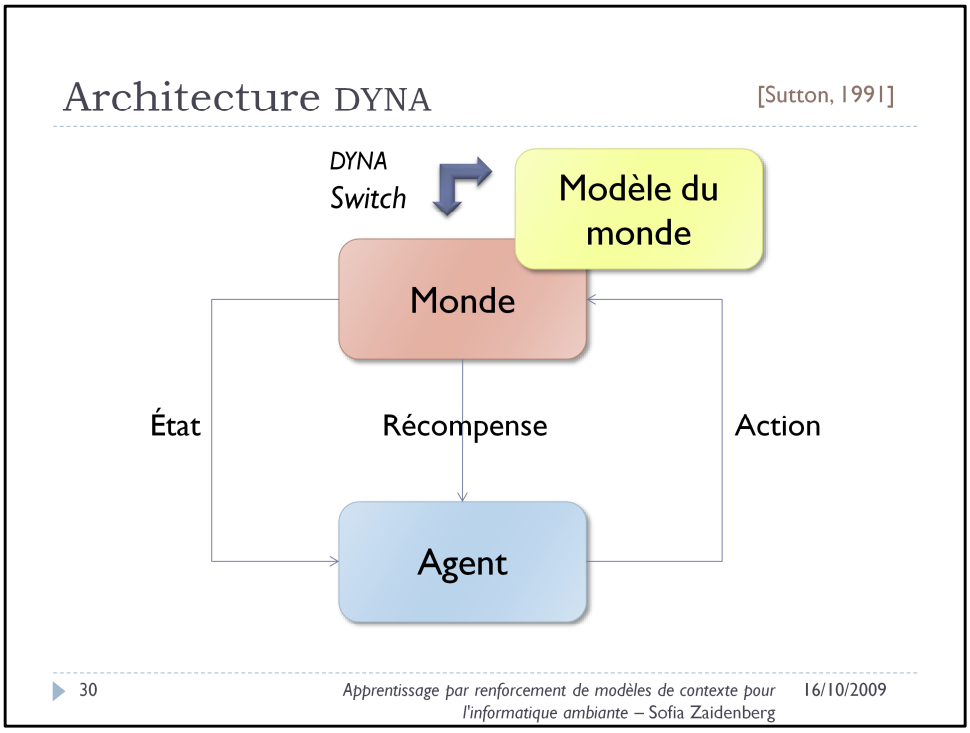
Long car beaucoup d'états, il faut expérimenter chaque action dans chaque état, donc beaucoup d'actions inappropriées (l'exploration n'est pas acceptable car l'utilisateur est dans la boucle).



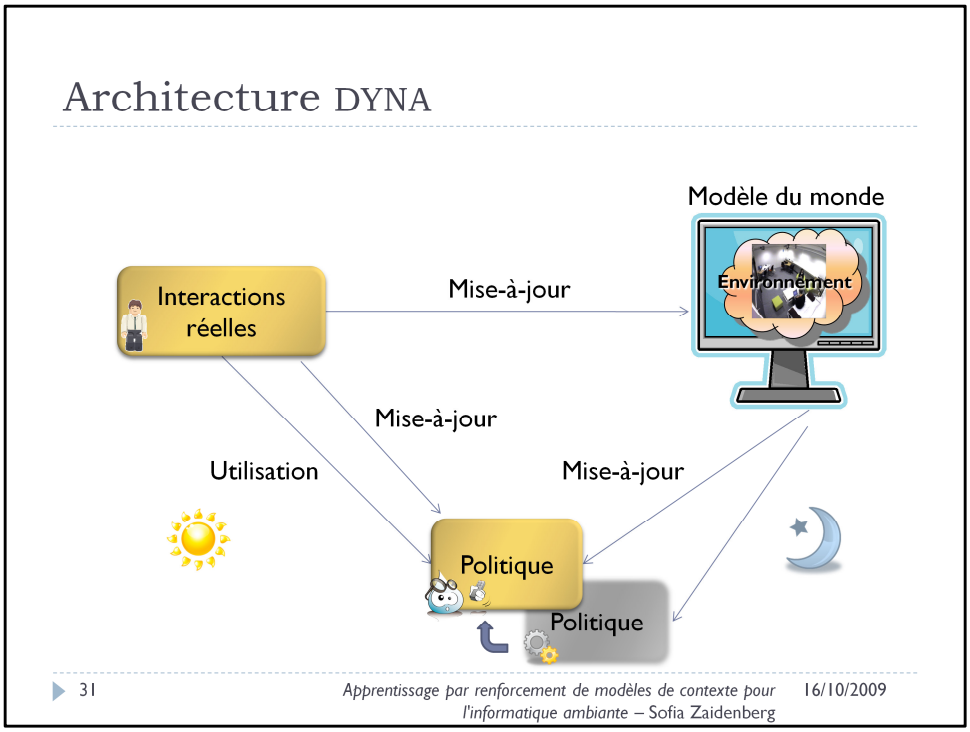
On voudrait revivre les expériences virtuellement, au lieu de les vivre dans le monde réel.
→Pour ça, on a besoin d'un modèle du monde réel.

On veut tirer le maximum de profit de chaque expérience.

Processus intrinsèquement exploratoire. Le système ne connait pas le monde et il va tâtonner. On ne raisonne pas. Le modèle permet de faire une phase de raisonnement hors ligne.
L'exploration est faite dans le modèle. Revivre l'expérience pour comprendre que c'est pas bien d'aller vite, et puis aller plus loin (explorer). Essayer de l'arrêter de plus en plus loin de la porte, et voir que ça ne suffit pas à pouvoir s'arrêter et éviter le choc, donc apprendre que même si on est loin de la porte, aller vite est dangereux.
Exploration indépendante des expériences vécues : faire l'action « activer interrupteur » dans le modèle et avoir une estimation des conséquences dès la 1ère fois qu'on est dans la même situation dans le monde réel.



Faire une partie de l'exploration dans le modèle, sans impliquer le monde réel.

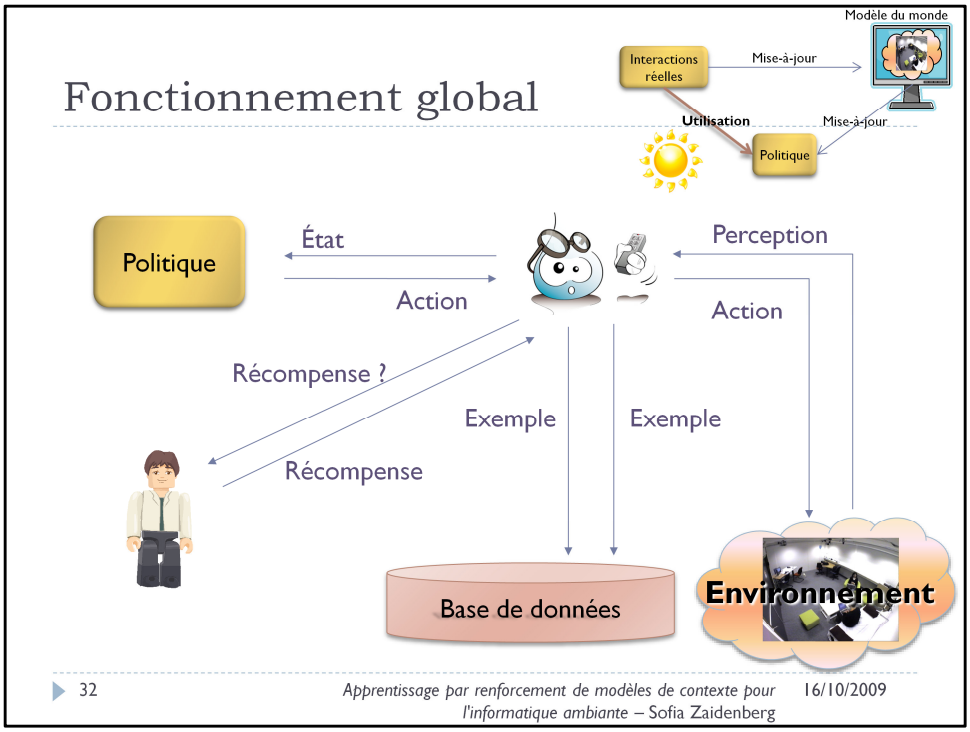


L'utilisateur qui interagit avec un système informatique (en général) construit inconsciemment un modèle mental de ce système.
Si le comportement du système change tout le temps, l'utilisateur n'arrivera pas à construire son modèle.
Un système dont on n'arrive pas à prédire les actions, qui surprend l'utilisateur, nuit à la confiance.

Donc on peut faire une 2ème politique interne (invisible à l'utilisateur), qui sera apprise en utilisant le modèle, sans impliquer l'utilisateur. La politique utilisée lors des interactions n'est pas modifiée sans prévenir l'utilisateur. On substitue la politique interne (nouvellement apprise) à la politique vue par l'utilisateur à certains moments bien définis (option de l'assistant : substitution tous les jours / toutes les semaines / à la demande, etc.).

Jour = fonctionnement interactif

Nuit = fonctionnement non interactif

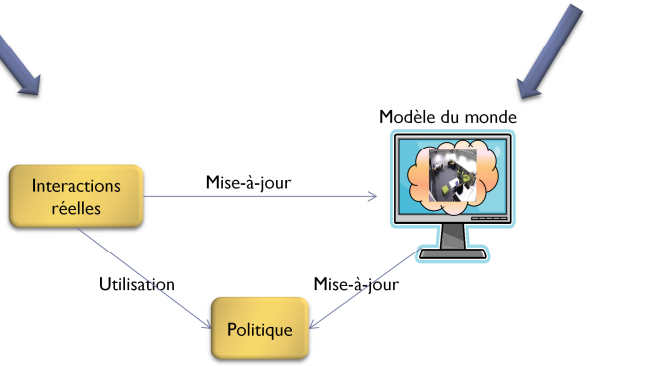


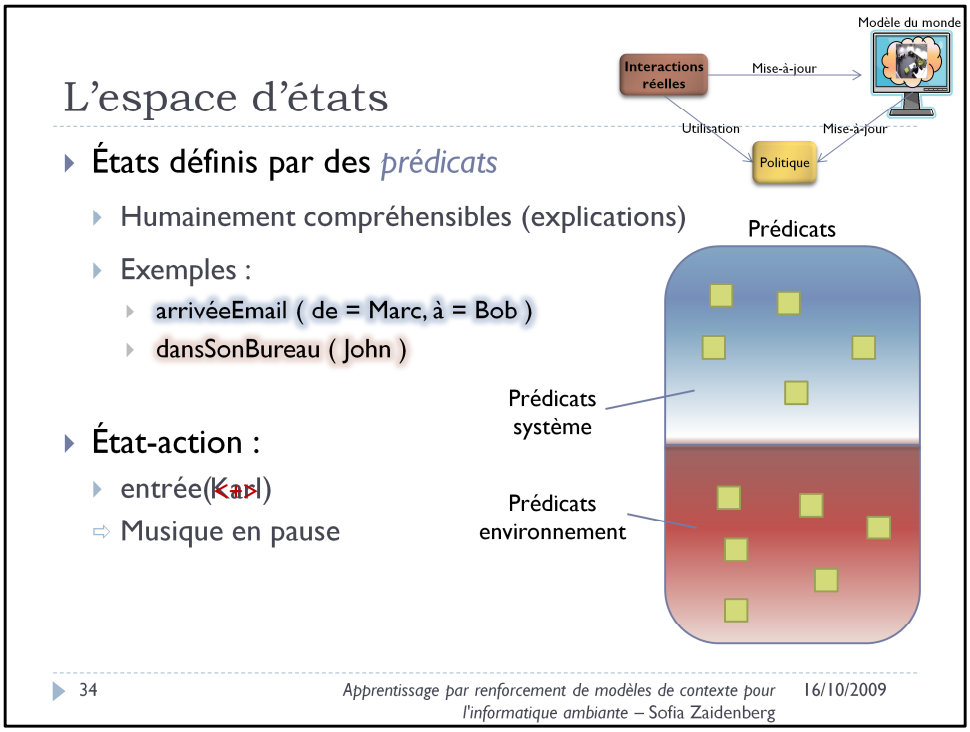
Exemples serviront à construire modèle du monde.

Partie interactive, sans apprentissage

Modélisation du problème

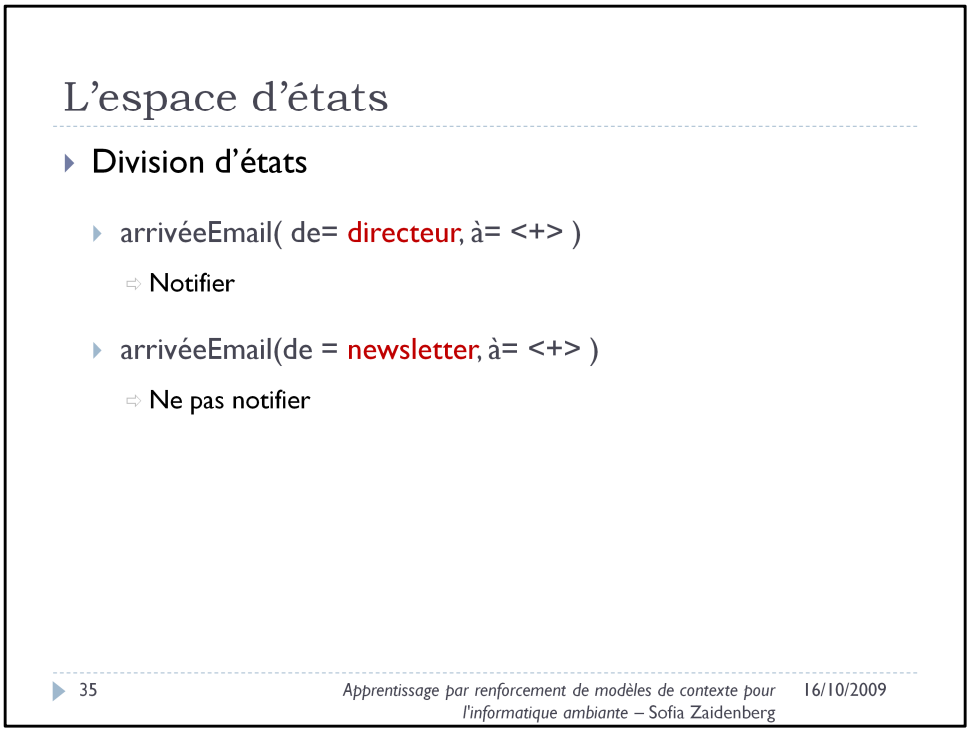
- Composants :
  - États
  - Actions
- Composants :
  - Modèle de transition
  - Modèle de récompense



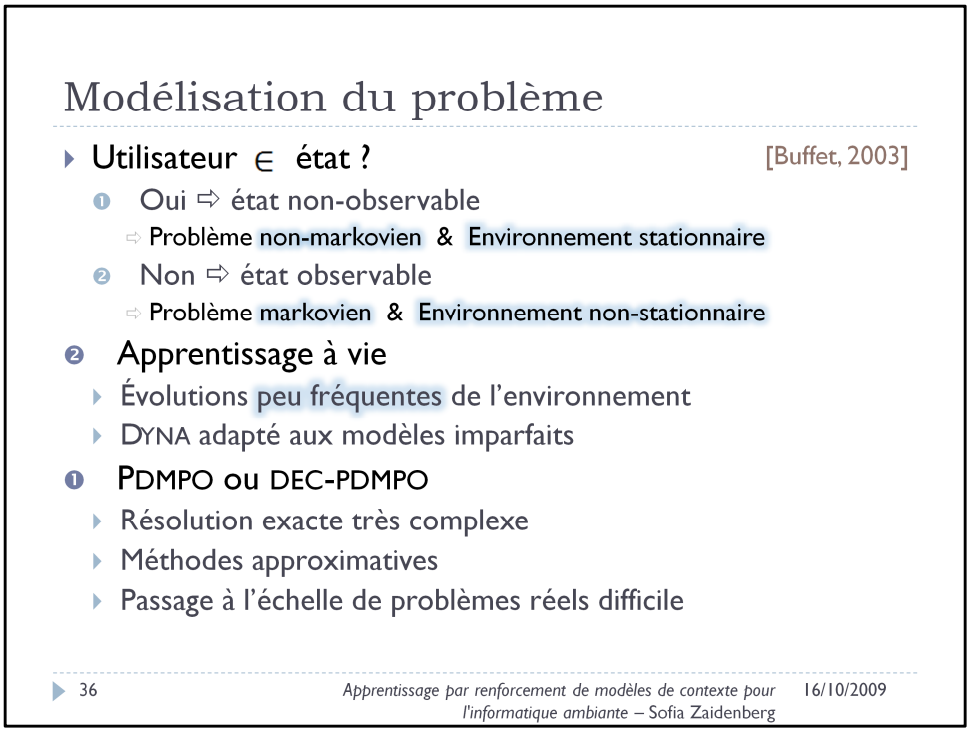


Prédicats du 1er ordre.

On peut pas être trop spécifique car trop d'états, on cherche à généraliser. Factoriser (généraliser) plus tôt permet d'avoir rapidement un comportement, sinon l'apprentissage n'arriverait pas à converger. Ça permet de tirer le max de profit de chaque expérience.



Dans les cas où la factorisation n'est pas pertinente (nuit à la satisfaction de l'utilisateur vis-à-vis du service), on fait une division des états factorisés a postériori (post-traitement).



En AR classique, l'état de l'agent est modifié seulement par actions de l'agent.
Chez nous, l'état est modifié par les actions mais aussi pas des événements extérieurs ou l'utilisateur (réception d'un mail, etc.).
On a défini l'état avec des éléments qu'on peut observer, est-ce qu'en plus on y incorpore l'utilisateur ?

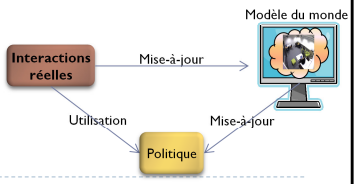
La non stationnarité provient des changements des préférences utilisateur.
Non stationnarité = conséquences des actions changent dans le temps → conséquence : je peux plus me comporter de la même façon, donc une action qui était bonne avant, ne l'est plus (les renforcements changent).
Ex : l'utilisateur aime bien lire son mail le matin, donc ouvrir son client mail le matin était une bonne action, puis il a changé ses habitudes et lit son mail l'après-midi, donc la même action (ouvrir client mail le matin) a une mauvaise conséquence.
Évolution monde provoquée par changement habitudes utilisateur.

Question : peut-on appliquer un apprentissage classique (comme dans le labyrinthe) dans un environnement non-stationnaire (imaginons que les obstacles du labyrinthe changent dans le temps…) ?
La réponse est oui, on peut car on suppose que utilisateur évolue lentement. L'environnement n'est pas stationnaire, mais il est quasi-stationnaire, il est localement stationnaire. Il est stationnaire pendant suffisamment longtemps pour qu'on puisse apprendre le bon comportement. Lorsqu'il change, on va le suivre. Par conséquent, on peut se ramener à un PDM.

On veut pas rentrer dans la problématique des PDMPO car c'est un problème trop complexe, on préfère rester dans le cadre mieux maitrisé des PDM.

L'espace d'actions

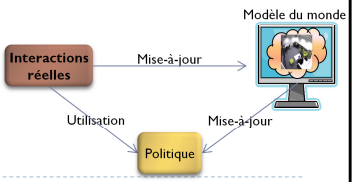
- Les actions possibles combinent
  - Transmettre un rappel à l'utilisateur
  - Informer d'un nouvel email
  - Verrouiller l'écran d'un ordinateur
  - Déverrouiller l'écran d'un ordinateur
  - Pauser la musique jouant sur un ordinateur
  - Relancer la musique jouant sur un ordinateur
  - Ne rien faire



Récompenses

- Récompenses explicites
  - Par une interface non intrusive

- Récompenses implicites
  - Collectées à partir d'indices (valeur numérique moindre)



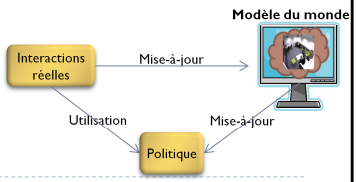
Récompenses données par l'utilisateur, ça pose différents problèmes (abordés dans le manuscrit).

Modèle de l'environnement

- Construits par apprentissage supervisé
  - À partir d'exemples réels
- Initialisés par le sens commun
  - Système fonctionnel immédiatement
  - Modèle initial *vs.* Q-valeurs initiales [Kaelbling, 2004]
  - Extensibilité

Modèle de transition

Modèle de récompense

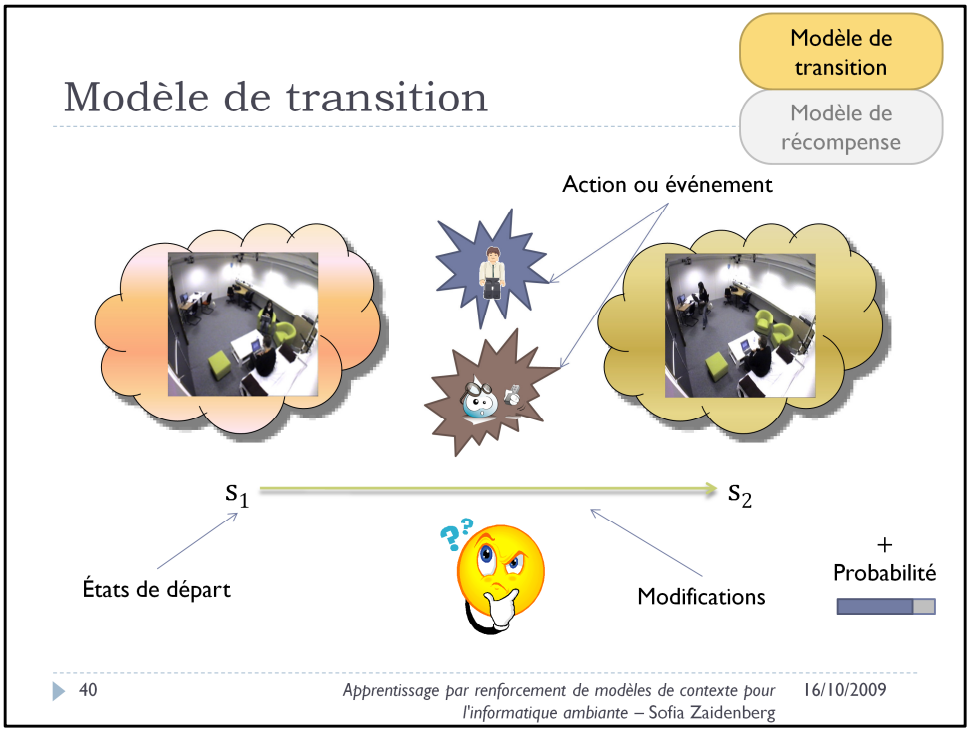


On collecte en permanence les données réelles pour améliorer et affiner le modèle.

Au premier lancement : on exécute AR en tâche de fond pour initialiser un comportement par défaut à partir des modèles par défaut.

On donne des modèles initiaux et pas des valeurs de qualité initiales car il est plus facile de spécifier un renforcement qu'un comportement [Kaelbling, 2004].

Modèle initial pourrait être utilisé pour initialiser un autre système d'apprentissage que l'AR, alors que la Q-table est propre à l'AR.



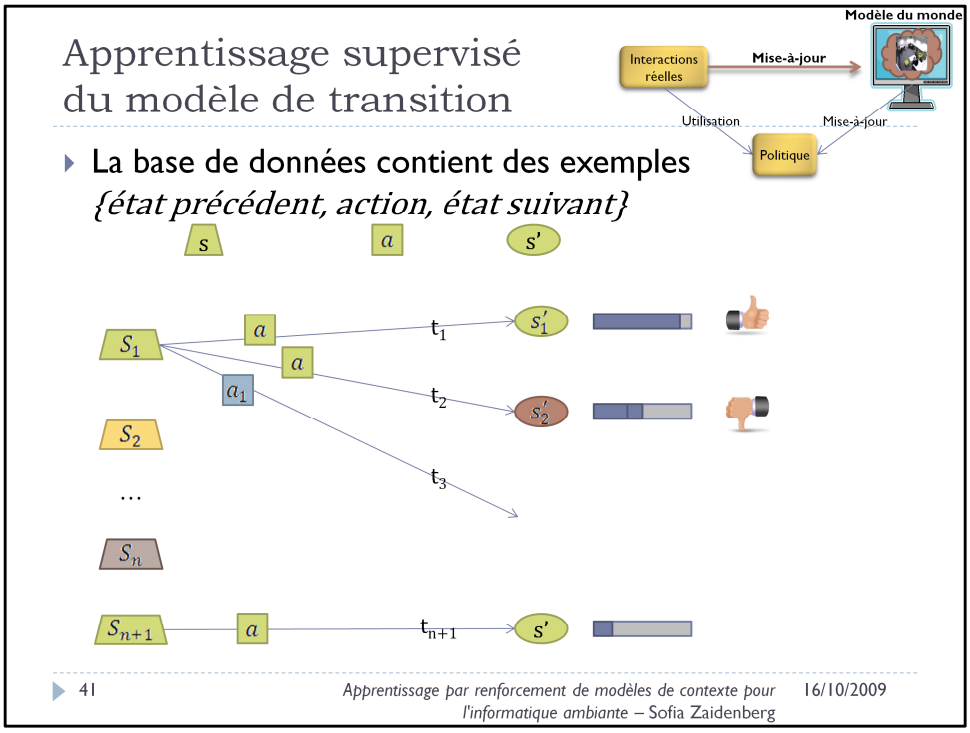
On veut apprendre ça mais pas trop précisément, en généralisant.
On veut comprendre comment ça se fait, le modéliser, on va calculer des transformations. Chaque transformation opère sur un état de départ factorisé.

Généralisation pour tirer le max de profit de chaque expérience, et pour permettre exploration dans le modèle plus tard.

Observation du monde n'est pas parfaite donc une même action peut donner des s2 différents --> probabilités (pour tenir compte de l'incertitude).

Modèle de transition = ensemble de transformations d'un état (factorisé) vers le suivant étant donnée une action ou un événement.
Une transformation est composée de

- Un état précédent
- Des modifications
- L'action (ou l'événement)
- Une probabilité

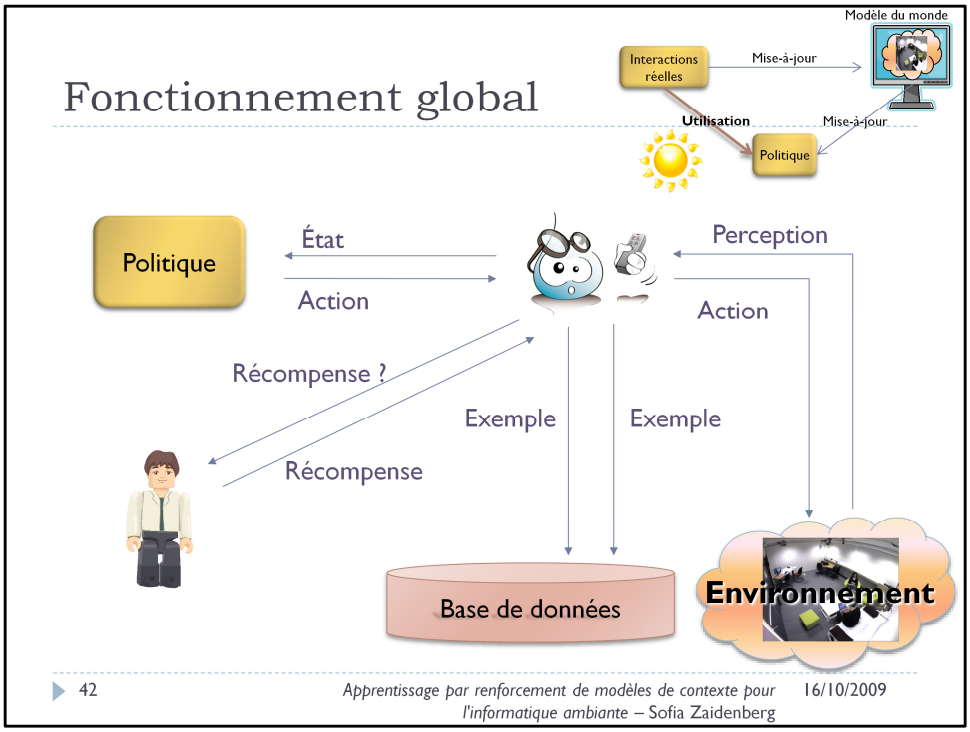


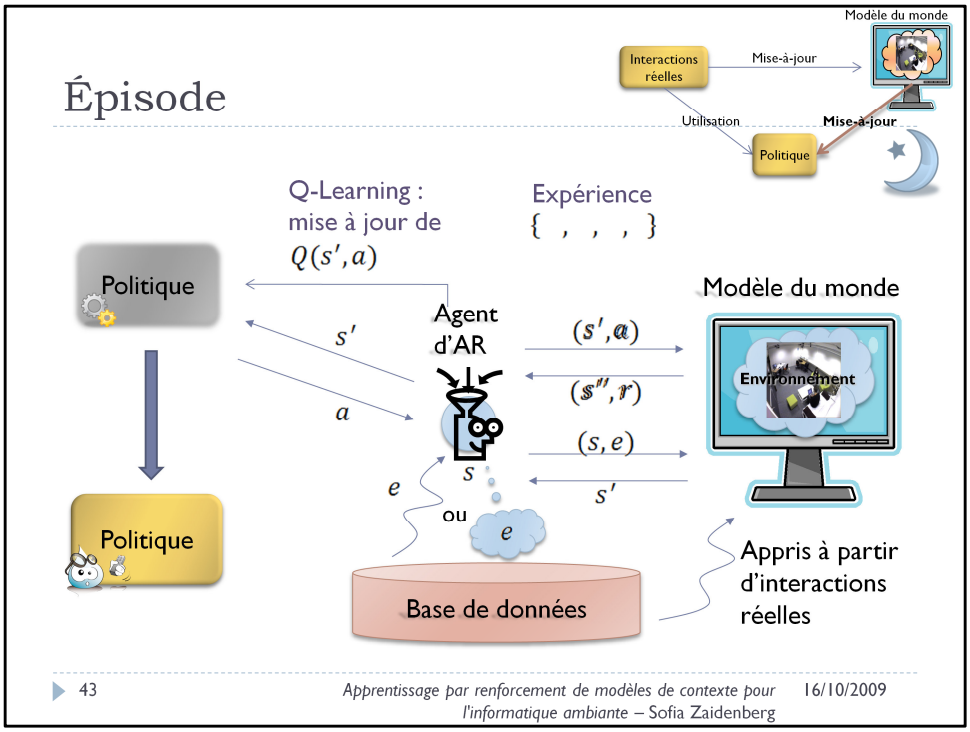
L'exemple renforce t1 et affaiblit t2, on cherche pas à stabiliser ces distributions car l'environnement est non stationnaire donc on sait qu'il va évoluer, et les probabilités doivent pouvoir évoluer avec lui. On ne diminue pas le poids des nouveaux exemples.

Nouvel exemple ⇨ nouvelle transformation générique pour faire de la généralisation.

Apprentissage « stable » : pour qu'un changement ait lieu dans le modèle, il faut voir l'exemple plusieurs fois (éviter d'apprendre les erreurs de capteurs). Si c'est un vrai changement, on aura forcément plusieurs exemples.

Fonctionnement global





Partie non interactive, apprentissage

1 cycle = 1 itération

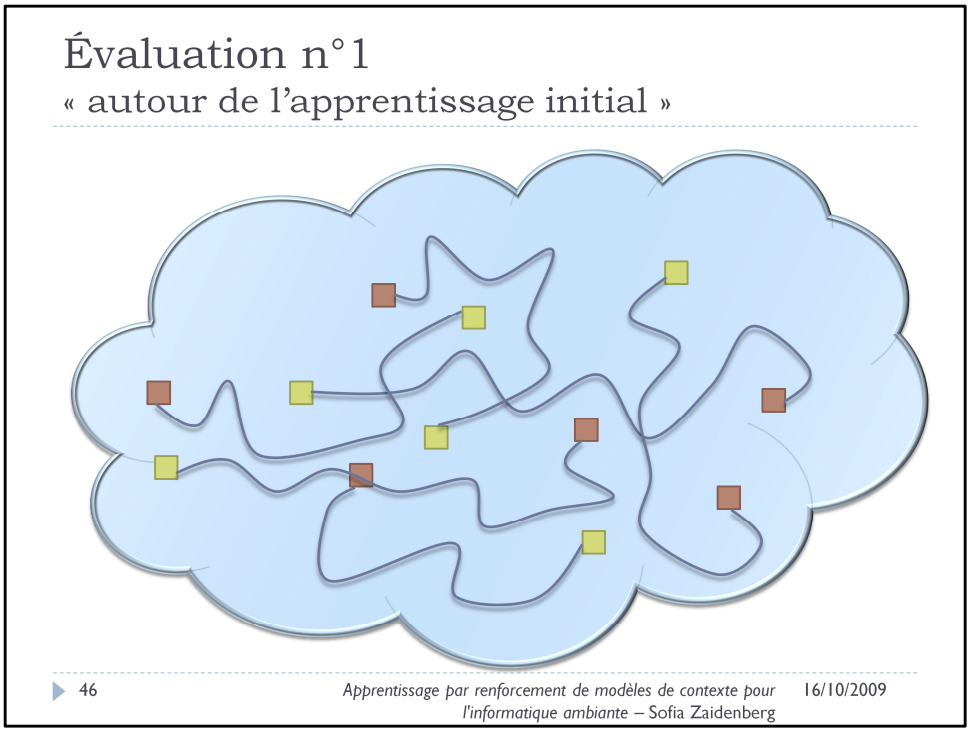
k itérations = 1 épisode

Plan

- Présentation du problème
- Apprentissage dans les systèmes ubiquitaires
- Enquête grand public
- Réalisation d'un système ubiquitaire
- Apprentissage par renforcement du modèle de contexte
- **Expérimentations et résultats**
- Conclusion

Expérimentations

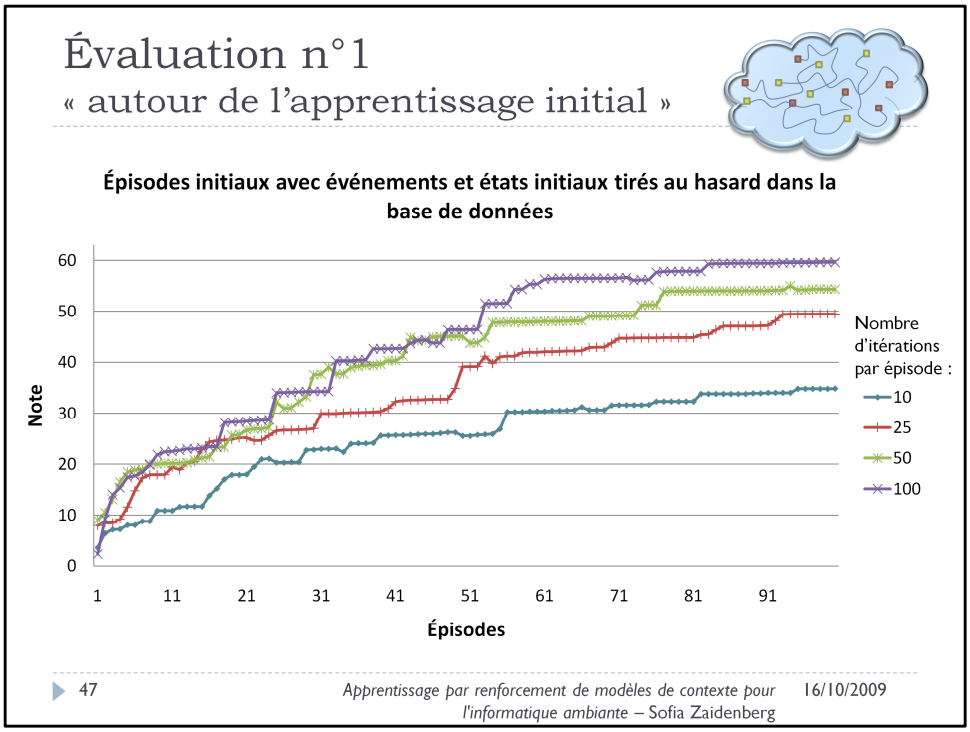
- Enquête grand public → évaluation qualitative
- Évaluations quantitatives en 2 étapes :
  - Évaluation de la phase initiale
  - Évaluation du système en fonctionnement normal



Supervisé : généralisation pas mémorisation d'exemples → permet exploration

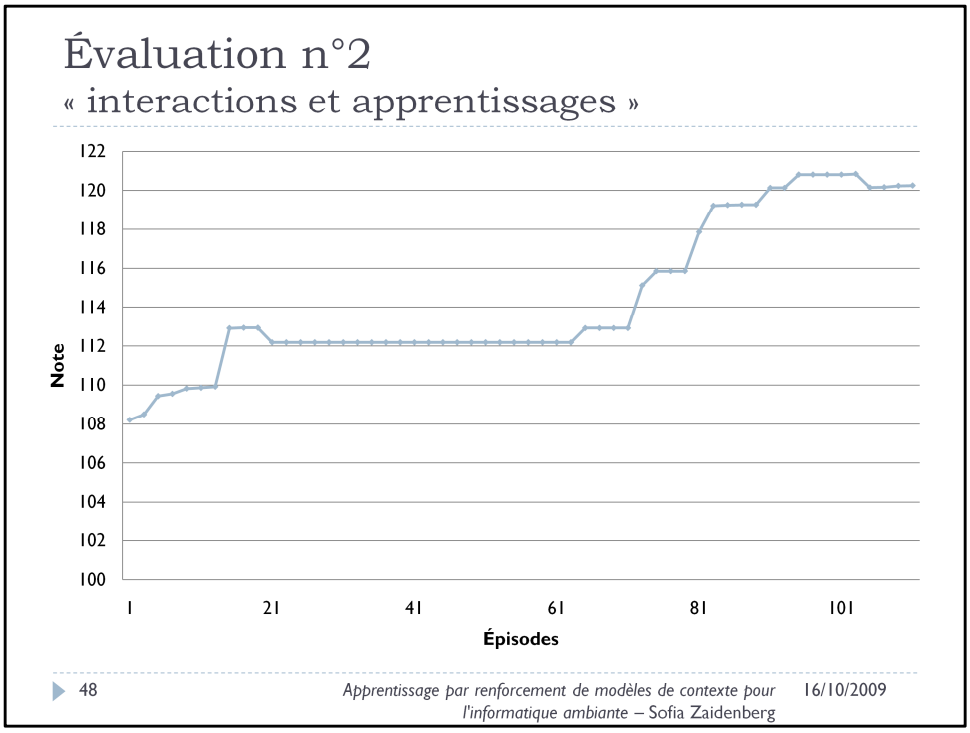
On a le modèle du monde initial, on veut l'explorer pour le traduire en un comportement initial. Il y a 4 paramètres à fixer : comment choisir les états de départ (carrés verts), comment choisir l'événement à chaque pas du chemin, combien de chemins faire, quelle longueur de chemins choisir ?
1 chemin = 1 épisode de k itérations.
Les 2 premiers ont été fixés dans le manuscrit (états de départ des chemins, et événements à chaque pas du chemin choisis aléatoirement parmi ceux déjà observés → permet d'obtenir un résultat optimal).
Pour la longueur et le nombre des chemins, voir le graphique suivant.

Exemple pour état de départ : l'état où Karl entre dans le bureau.



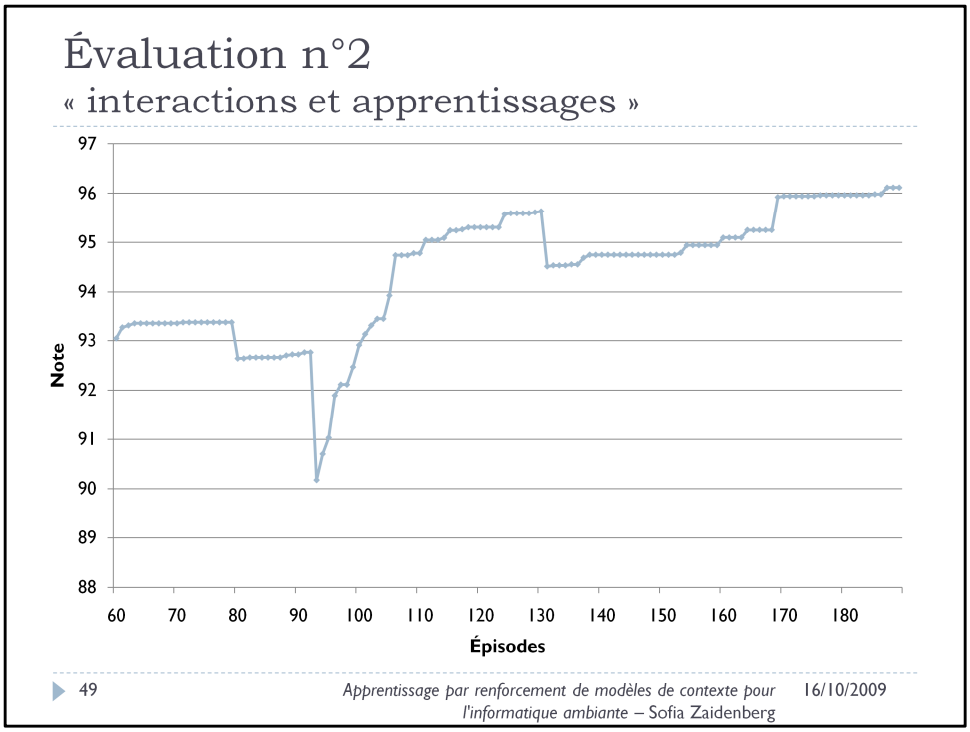
En abscisse : le nombre de chemins, en ordonnée : une note exprimant la ressemblance du comportement appris au comportement désiré (on a cette note car c'est une expérience et l'expérimentateur a donné des notes, dans la « vie réelle », la note n'est pas disponible). Les différentes courbes correspondent à différentes longueurs de chemins.

Dans l'absolu, on devrait choisir le dernier point de la courbe violette (100 itérations/épisode), mais dans la pratique on devrait choisir un compromis entre temps d'exécution et qualité du résultat.
Entre la courbe verte et la violette, on double le temps de calcul, et ce doublement n'est pas justifié par la si faible augmentation en termes de qualité de comportement. De même, à partir de l'épisode 50, on ne gagne pas grand-chose au niveau de la note, alors que le temps de calcul augmente toujours autant. Donc on pourra choisir, pour cette phase initiale, d'exécuter 50 épisodes de 50 itérations chacun.



Point de départ = comportement initial.
Puis l'expérimentateur interagit avec l'environnement. Ceci permet à l'assistant de voir des parties du monde non incluses dans le modèle initial, et donc l'apprendre. Là où la courbe est plate, l'assistant a appris tout ce qu'il pouvait, il n'observe rien de nouveau. Puis, dès qu'il observe une nouveauté, il continue à apprendre.

L'expérimentateur utilise le tableau de bord (cf backup slides) pour interagir avec l'environnement, simplement pour lui faciliter l'expérience, mais on a toujours le facteur humain dans l'expérience.



Là, l'expérimentateur a changé d'avis (vers x=100) → la note a baissé avant de remonter (exemple d'évolution de l'environnement).

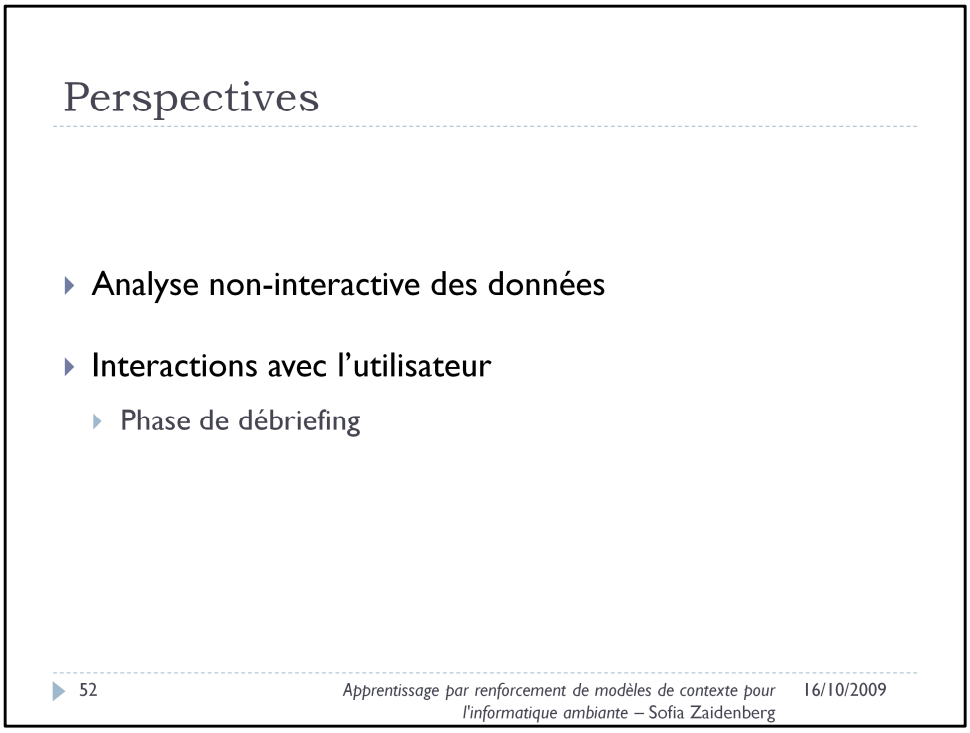
Par exemple, l'utilisateur ne veut plus lire ses mails le matin, mais l'après-midi.

Plan

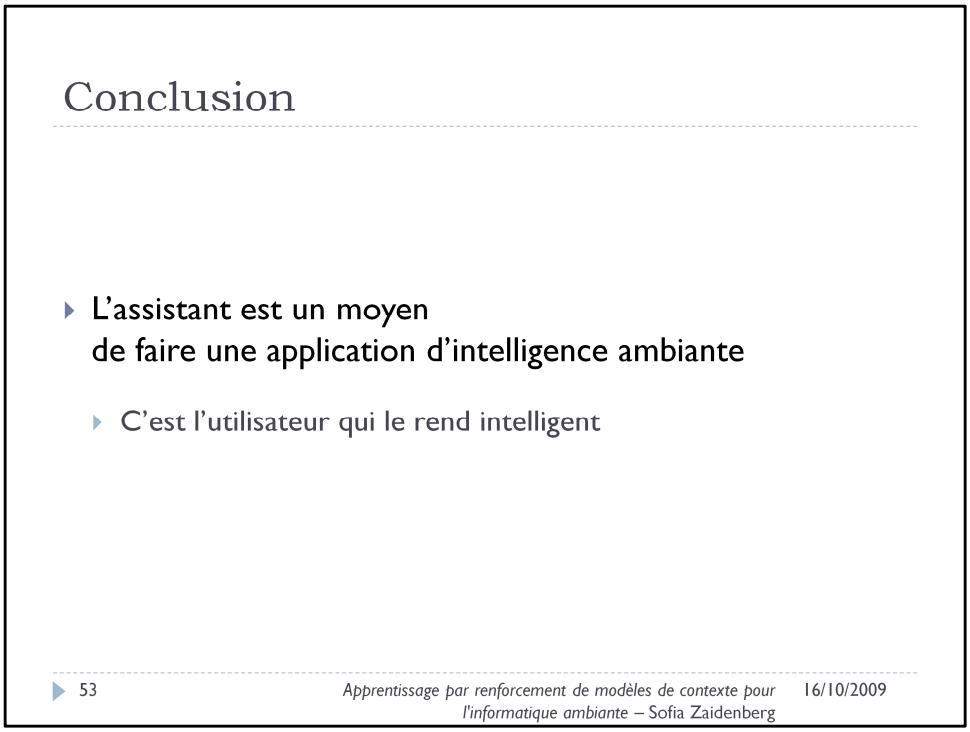
- Présentation du problème
- Apprentissage dans les systèmes ubiquitaires
- Enquête grand public
- Réalisation d'un système ubiquitaire
- Apprentissage par renforcement du modèle de contexte
- Expérimentations et résultats
- **Conclusion**

Contributions

- Personnalisation d'un système ubiquitaire
  - Sans spécification explicite
  - Évolutive
- Adaptation de l'apprentissage par renforcement indirect à un problème réel
  - Construction d'un modèle du monde
  - Injection de connaissances initiales
- Mise en place d'un prototype



Interactions : on n'est peut-être pas obligés de toujours tout découvrir par exploration. C'est plus efficace pour nous de pouvoir parfois demander à l'utilisateur, et lui, il préfère aussi parfois pouvoir donner son avis (cf enquête).



L'assistant est une boite blanche, il n'est pas intelligent par sa conception, ce sont ses interactions avec l'utilisateur, les retours qu'il donne, qui rendent l'assistant intelligent. C'est l'engagement de l'utilisateur dans le système qui lui donne de la valeur ajoutée.

Merci de votre attention

Questions ?

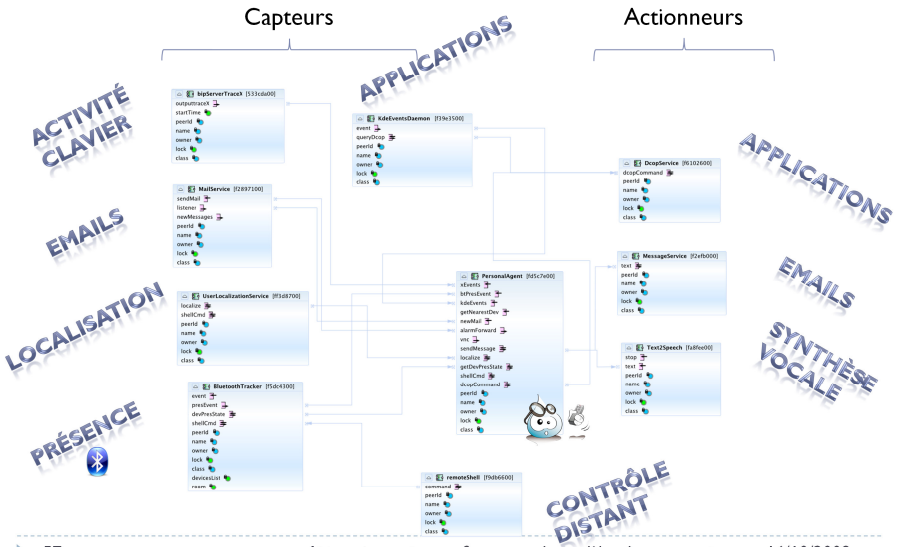
Bibliographie

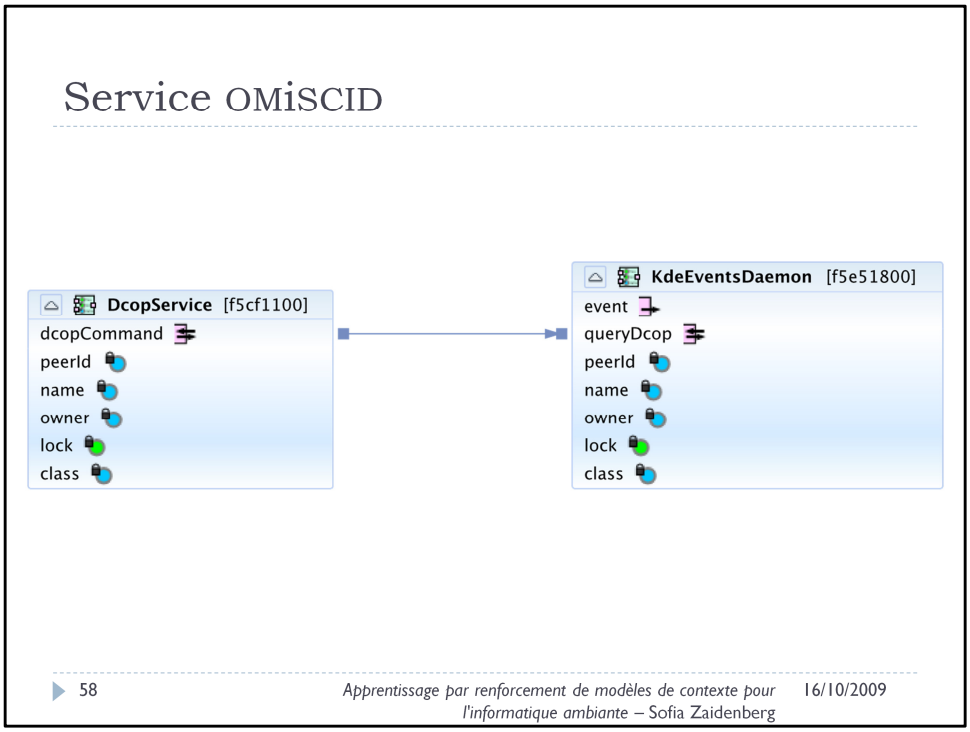
[Bellotti et Edwards, 2001]	Victoria BELLOTTI et Keith EDWARDS. « Intelligibility and accountability: human considerations in context-aware systems ». *Dans Human-Computer Interaction*, 2001.
[Brdiczka *et al.*, 2007]	Oliver BRDICZKA, James L. CROWLEY et Patrick REIGNIER. « Learning Situation Models for Providing Context-Aware Services ». *Dans Proceedings of HCI International*, 2007.
[Buffet, 2003]	Olivier Buffet. « Une double approche modulaire de l'apprentissage par renforcement pour des agents intelligents adaptatifs ». Thèse de doctorat, Université Henri Poincaré, 2003.
[Emonet *et al.*, 2006]	Rémi Emonet, Dominique Vaufreydaz, Patrick Reignier et Julien Letessier. « O3MiSCID: an Object Oriented Opensource Middleware for Service Connection, Introspection and Discovery ». *Dans 1st IEEE International Workshop on Services Integration in Pervasive Environments*, 2006.
[Kaelbling, 2004]	Leslie Pack Kaelbling. « Life-Sized Learning ». Lecture at CSE Colloquia, 2004.
[Maes, 1994]	Pattie MAES. « Agents that reduce work and information overload ». *Dans Commun. ACM*, 1994.
[Maisonnasse 2007]	Jerome MAISONNASSE, Nicolas GOURIER, Patrick REIGNIER et James L. CROWLEY. « Machine awareness of attention for non-disruptive services ». *Dans HCI International*, 2007.
[Moore, 1975]	Gordon E. MOORE. « Progress in digital integrated electronics ». *Dans Proc. IEEE International Electron Devices Meeting*, 1975.

Bibliographie

[Nonogaki et Ueda, 1991]	Hajime Nonogaki et Hirotada Ueda. « FRIEND21 project: a construction of 21st century human interface ». *Dans CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, 1991.
[Roman *et al.*, 2002]	Manuel ROMAN, Christopher K. HESS, Renato CERQUEIRA, Anand RANGANATHAN, Roy H. CAMPBELL et Klara NAHRSTEDT. « Gaia: A Middleware Infrastructure to Enable Active Spaces ». *Dans IEEE Pervasive Computing*, 2002.
[Sutton, 1991]	Richard S. Sutton. « Dyna, an integrated architecture for learning, planning, and reacting ». *Dans SIGART Bull,* 1991.
[Weiser, 1991]	Mark WEISER. « The computer for the 21st century ». *Dans Scientic American*, 1991.
[Weiser, 1994]	Mark WEISER. « Some computer science issues in ubiquitous computing ». *Dans Commun. ACM*, 1993.
[Weiser et Brown, 1996]	Mark WEISER et John Seely BROWN. « The coming age of calm technology ». http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm, 1996.
[Watkins, 1989]	CJCH Watkins. « Learning from Delayed Rewards ». Thèse de doctorat, University of Cambridge, 1989.

Interconnexion des modules





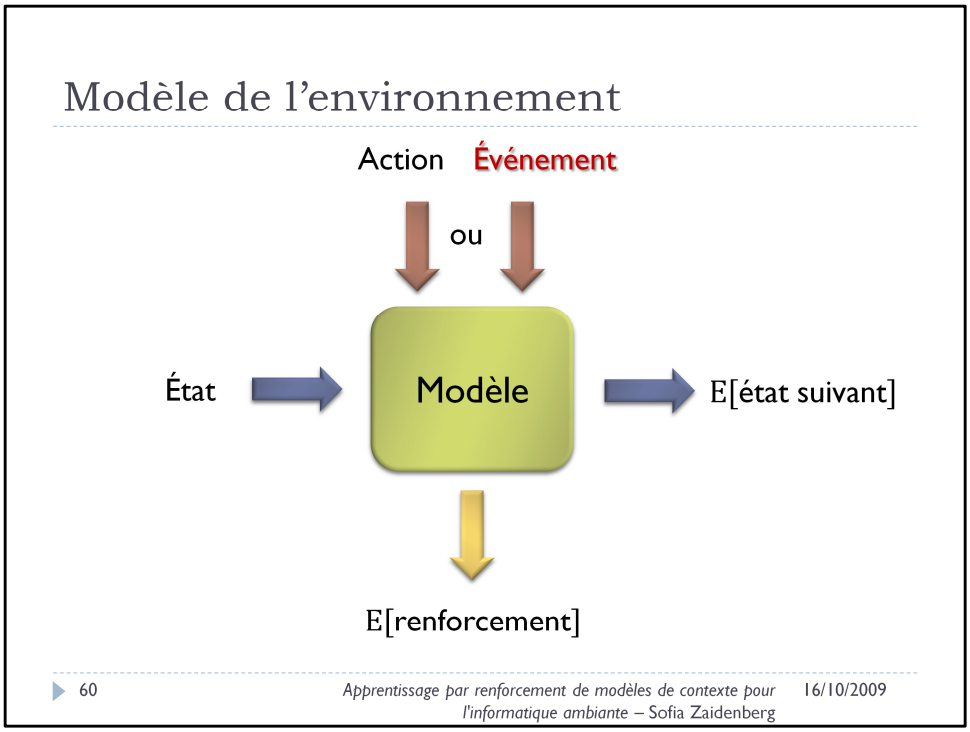
Les modules du système sont implémentés en tant que « services OMiSCID »
Ils exposent des connecteurs et des variables

- Entrée / sortie / entrée-sortie pour les connecteurs
- Lecture / lecture-écriture pour les variables

Deux services communiquent en « branchant » deux de leurs connecteurs

Définition d'un état

Prédicat	Arguments
alarm	title, hour, minute
xActivity	machine, isActive
inOffice	user, office
absent	user
hasUnreadMail	from, to, subject, body
entrance	isAlone, friendlyName, btAddress
exit	isAlone, friendlyName, btAddress
task	taskName
user	login
userOffice	office, login
userMachine	machine, login
computerState	machine, isScreenLocked, isMusicPaused



Normalement il n'y a que les actions qui modifient l'état. L'utilisateur ne fait pas partie du monde, donc n'est pas modélisé dans le modèle du monde, donc les événements qui sont provoqués par ses actions sont des entrées du modèle.

Réduction de l'espace d'états

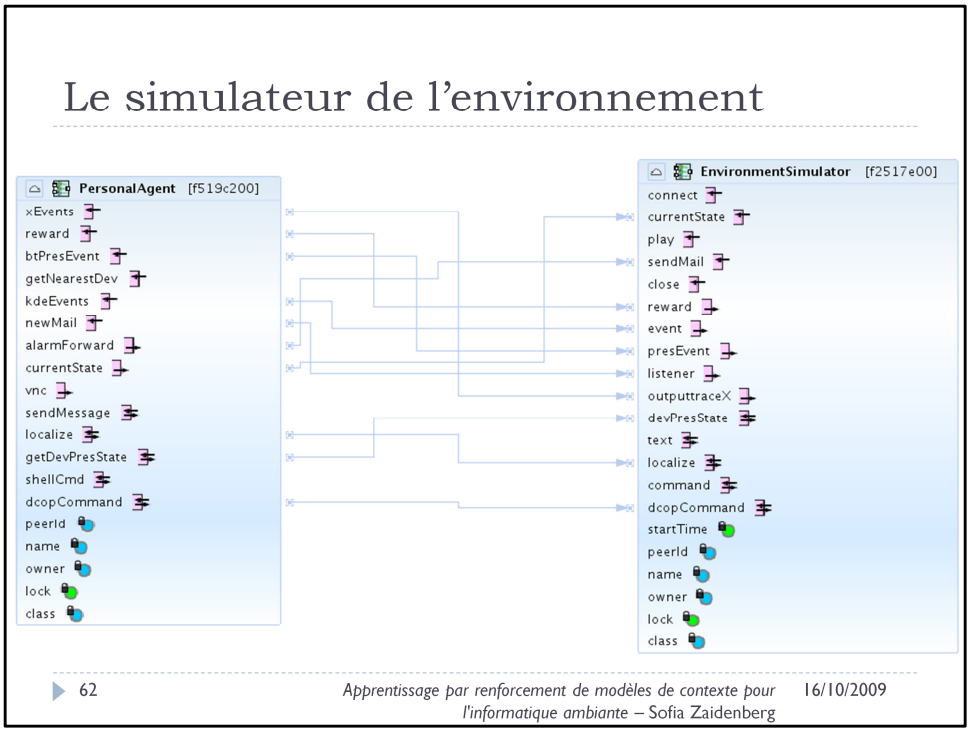
- Accélération de l'apprentissage
  - Factorisation d'états

Jokers
<*> et <+>

État	Action	Q-valeur
…entrance(isAlone=true, friendlyName=<+>, btAddress=<+>)…	pauseMusic	125.3

  - Division d'états

État	Action	Q-valeur
…hasUnreadMail(from=**boss**, to=<+>, subject=<+>, body=<+>)…	inform	144.02
…hasUnreadMail(from=**newsletter**, to=<+>, subject=<+>, body=<+>)…	notInform	105



Le simulateur contient une simulation de tous les capteurs et effecteurs en définissant les mêmes connecteurs qu'eux tous.
Les messages échangés sont de même format que sur les vrais connecteurs. Les échanges de messages suivent le même format (ex: requête – réponse ou juste requête etc.)

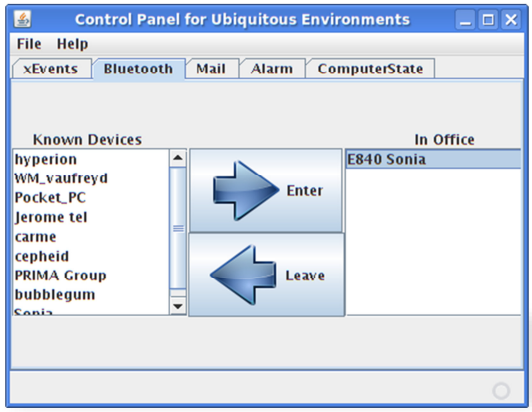
Critère d'évaluation : la note

- Résultat de l'AR : une Q-table
- Comment savoir si elle est « bonne » ?
- Apprentissage réussi si
  - Comportement correspond aux souhaits de l'utilisateur
  - Et c'est mieux si on a beaucoup exploré et si on a une estimation du comportement dans beaucoup d'états

$$note = \frac{1}{13}(10 \times n_{correct} + 2 \times p_{nonNul} + n_{total})$$

« Le tableau de bord »

- Permet d'envoyer par un clic les mêmes événements que les capteurs



Modèle de récompense

- Ensemble d'entrées spécifiant
  - Des contraintes sur certains arguments de l'état
  - Une action
  - La récompense

Modèle de récompense

Modèle de transition

Modèle de récompense

Action

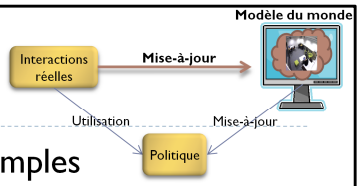
-50

Récompense

s_1

États de départ

Apprentissage supervisé du modèle de récompense



- La base de données contient des exemples *{état précédent, action, récompense}*