



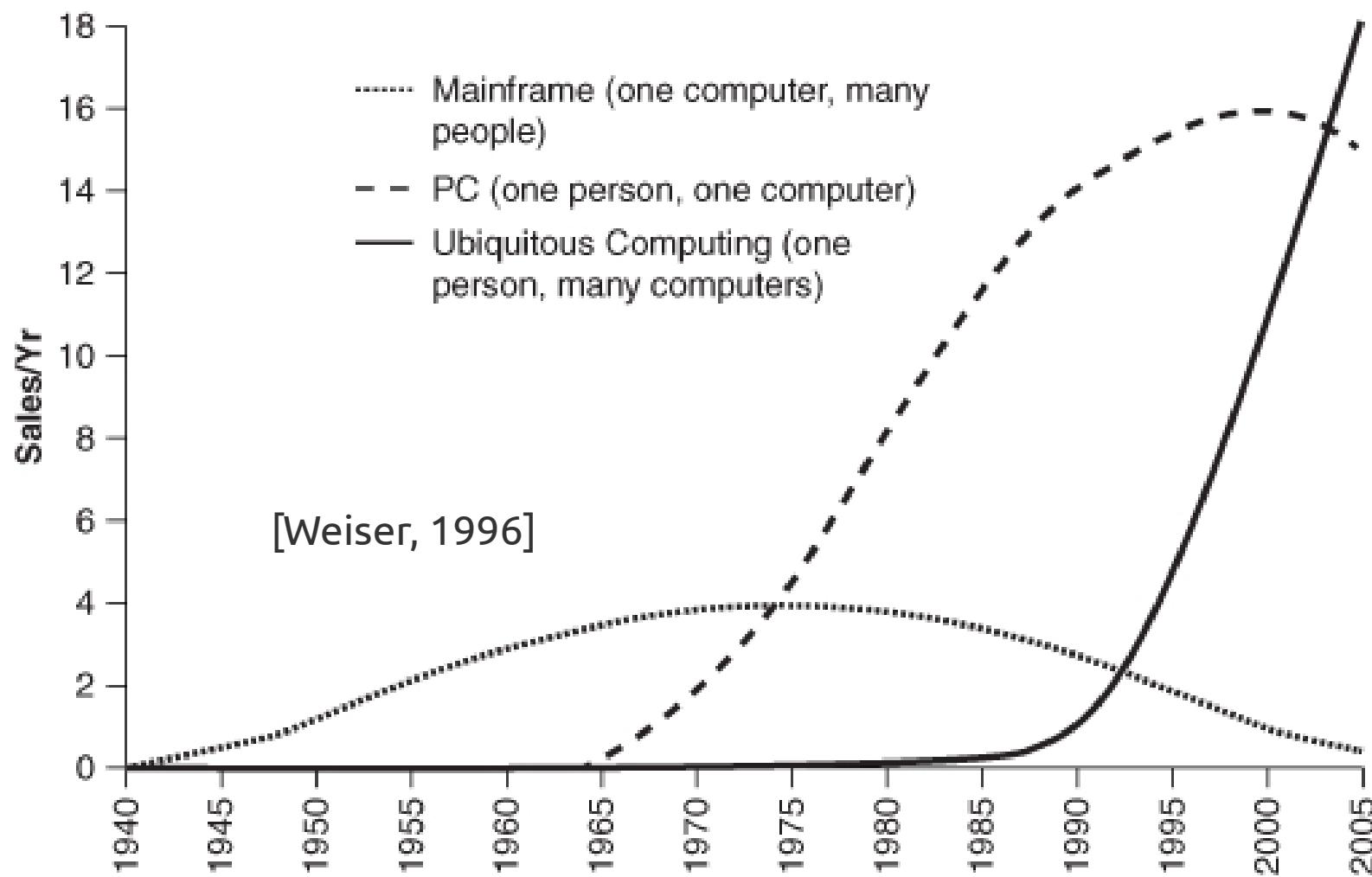
Systèmes Répartis, Déploiement et Mobilité du logiciel

SITA – 3A – v2016.1

Historique

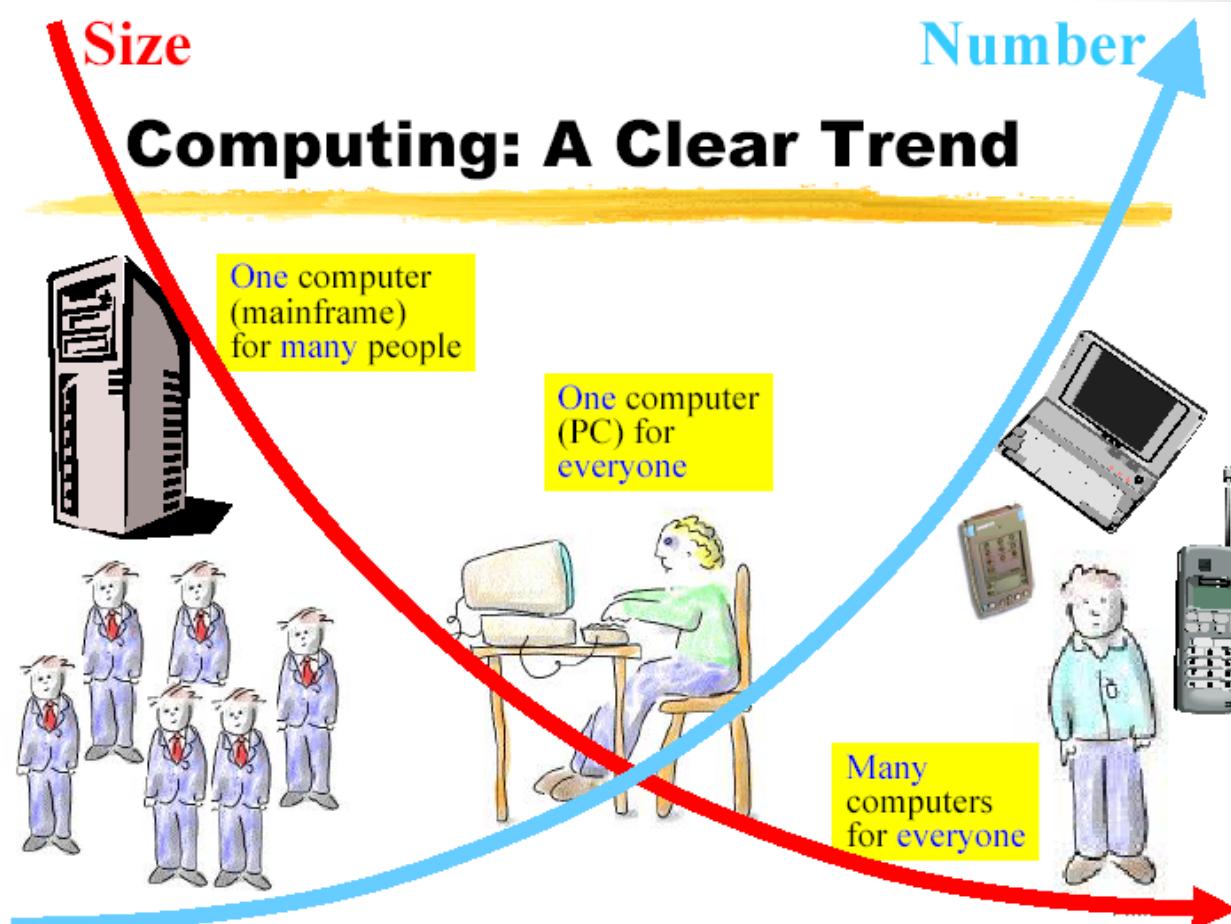


- Historique (<2000)



- [Mark Weiser, 1991]
 - « A new way of thinking about computers in the world, one that takes into account the natural human environment and allows the computers themselves to vanish in the background »
 - « The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it »

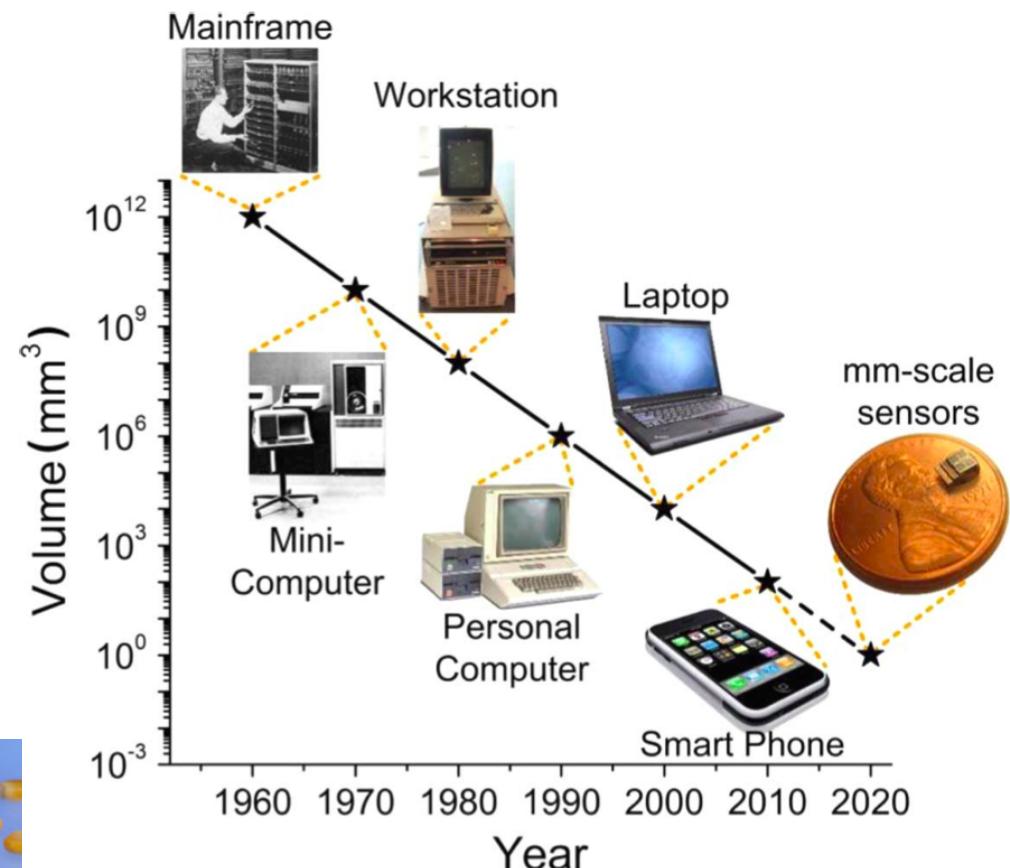
M. Weiser « The computer for the twenty-first century », Scientific American, sept 1991:94-104



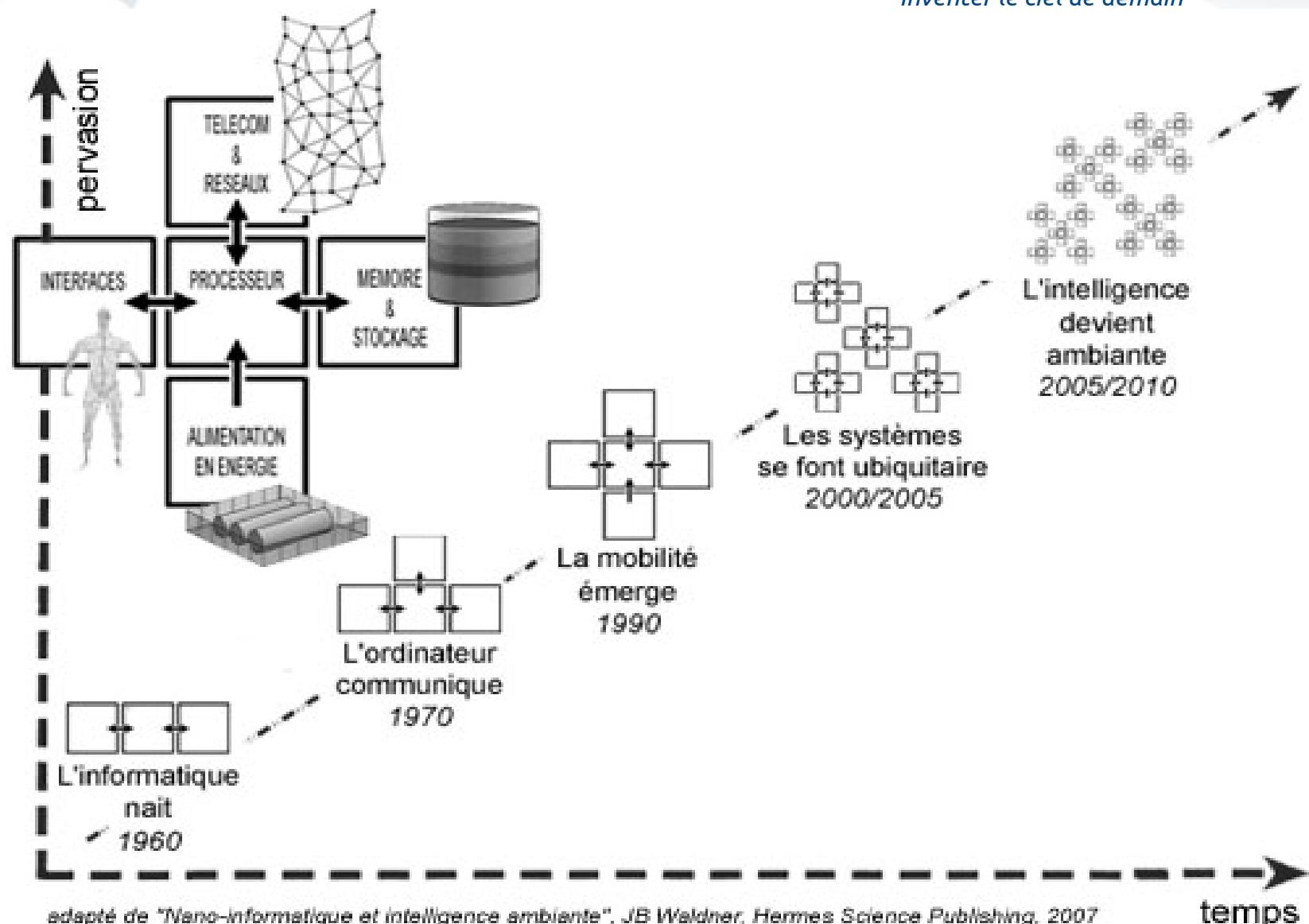
F. Mattern. « Ubiquitous & Pervasive Computing: A Technology-driven Motivation », Summer school on ubiquitous and pervasive computing, 2002

Historique

- Historique (>2000)
 - Loi de Moore...
 - Réduction du volume
 - Réduction des coûts
 - L'utilisateur n'a plus conscience d'être entouré par des smart-objets voire même de la *smart poussière* !



Historique



adapté de "Nano-informatique et intelligence ambiante", JB Waldner, Hermes Science Publishing, 2007

- Informatique ambiante ?



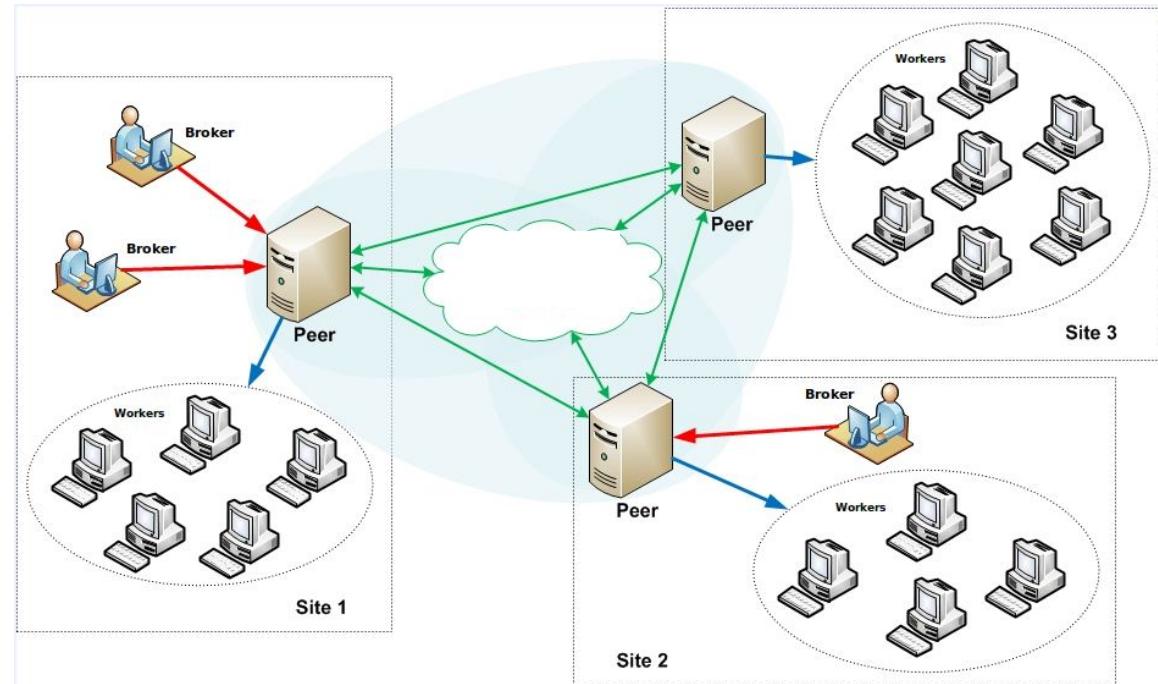
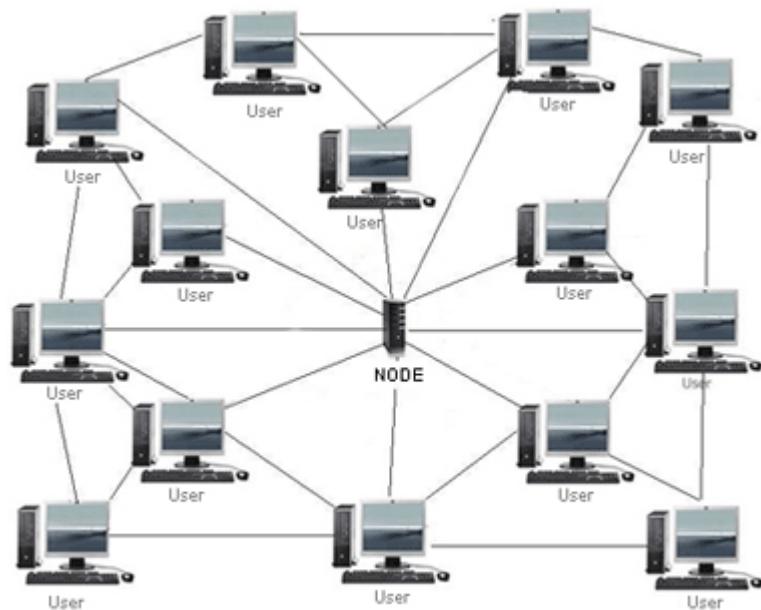
Historique (2)

- [M. Satyanarayanan, 2001]
 - Au delà des systèmes distribués et mobiles
- 4 challenges
 - Utilisation effective des « espaces intelligents »
 - Invisibilité
 - « Scalabilité » / passage à l'échelle
 - Transparence des inégalités d'environnements

M. Satyanarayanan « Pervasive computing : Visions and challenges », IEEE Personal Communications, aug. 2001:10-17

Vocabulaire

- P2P et Grid computing



Vers l'internet des objets...



- Connexion réseaux omniprésentes
 - Filaire, Wifi, Bluetooth, GPRS/GSM/3G...
- Périphériques « intelligents » (capacité de traitement et d'interaction)
 - Services élémentaires
- Disponibilité d'un point d'accès Internet
 - Interconnexion des systèmes entre eux
 - Accès à des services complexes
- Internet des objets ou des choses « Internet of things »

Les objets intelligents



- Peuvent se souvenir des événements importants
 - Mémoire
- Présentent un comportement dépendant du contexte
 - Capteurs
- Ils sont interactifs
 - Communiquent avec leur environnement
 - En réseau avec les autres objets intelligents
- Ce n'est plus de la science-fiction, ces objets existent (cf. « laboratoire informatique ambiante »)

La grande problématique



- Comment faire interagir des « objets intelligents »
 - Quand on n'est pas sûr de ce qu'il y aura à l'exécution
 - Quand ces objets peuvent tomber en panne (plus il y en a, plus la probabilité que l'un d'entre eux soit en panne augmente)
 - Quand on veut pouvoir profiter d'objets non prévus à la conception
 - ...
- Les objets ont beau être intelligents, ça nous fait une belle jambe si on ne sait pas construire de l'interaction ! Et ce n'est pas si facile...

Les « petites » problématiques



- Découverte
- Exploitation
- Gestion du contexte
- Hétérogénéité
- Administration décentralisée
- Volatilité
- Volume(s)
- Déploiement,
- Adaptation
- Orchestration...

La question « Bonus »

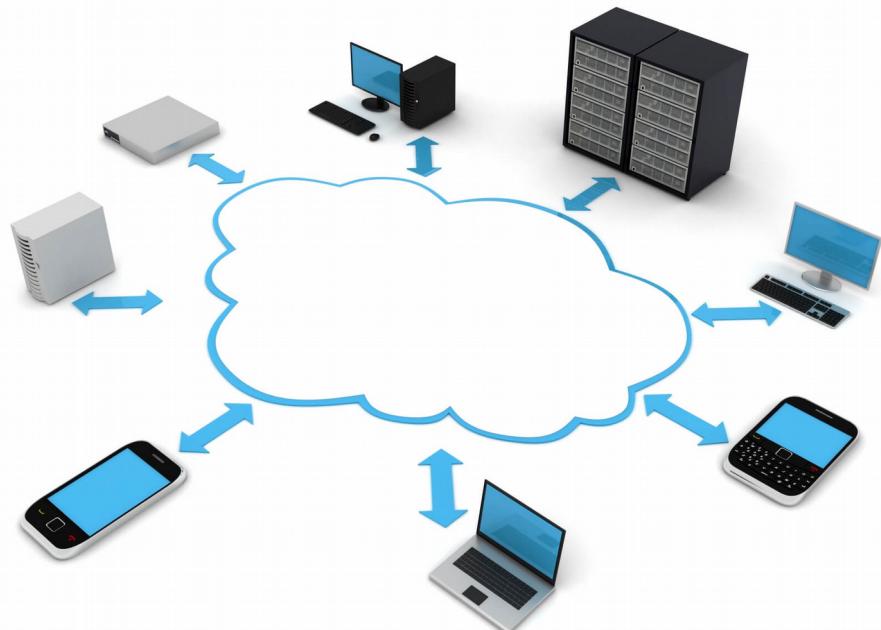


- Où doit-on exécuter le code ?
 - Faut-il exécuter le code au plus près des données/capteurs ou est-il préférable de le déporter sur une machine tierce ?
- Pas de réponse universelle, compromis entre :
 - Taille de code et mémoire disponible
 - Consommation électrique
 - Bande passante
 - Distance de communication
 - Environnement de programmation / langage
 - Besoins temps-réel
 - Exigences de sécurité

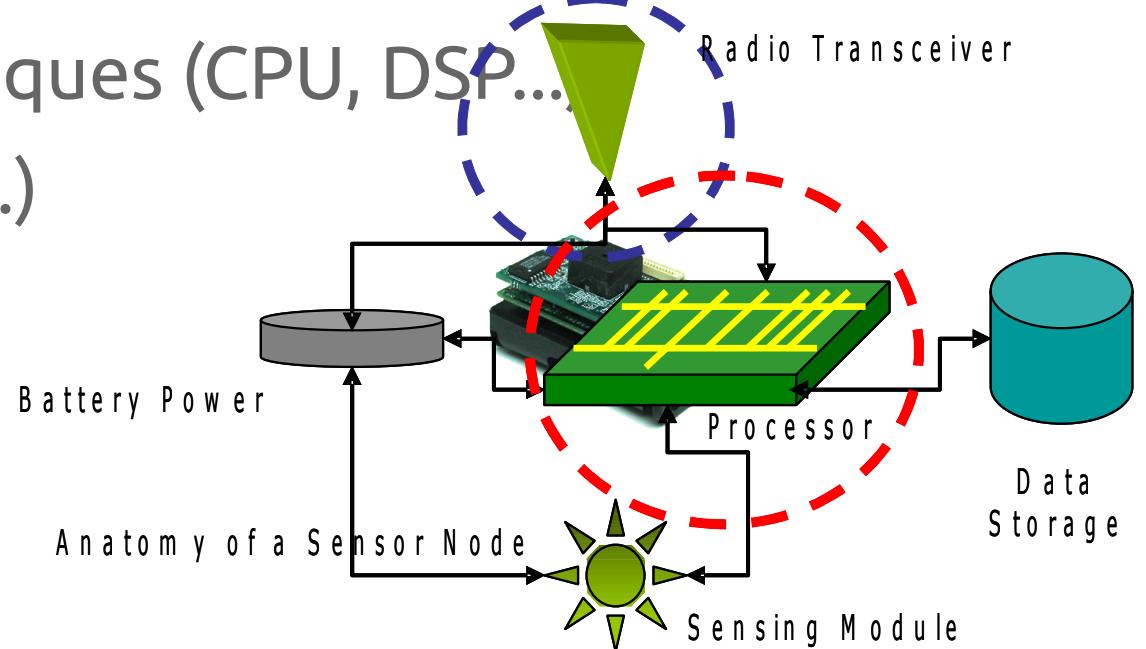
Vocabulaire



- IoT et Cloud computing



- Un capteur ou un effecteur a besoin d'un support d'exécution qui lui fournit :
 - Énergie (électrique, au moins aujourd'hui)
 - Moyen de communication (I/O série → Wifi...)
 - Processeurs logiques (CPU, DSP...)
 - Mémoire (RAM...)
 - Module capteur



Ex : ScatterWeb



- “Sensornetwork”
- Radio (zigbee -> ipv6lopan)
- Prix <100 €



II - Positionnement



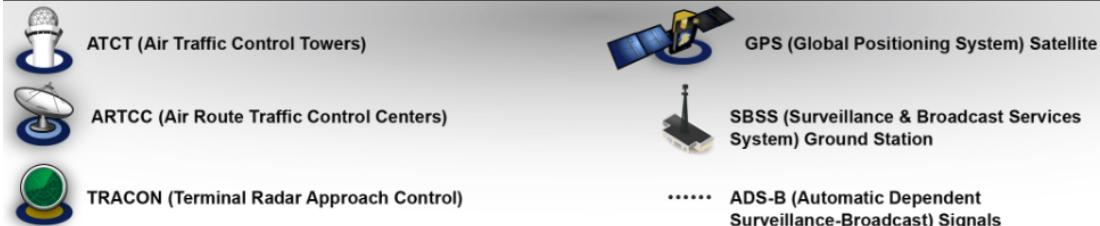
- Exemple : « smart city »



II - Positionnement



- Exemple : secteur ATM (contrôle aérien)



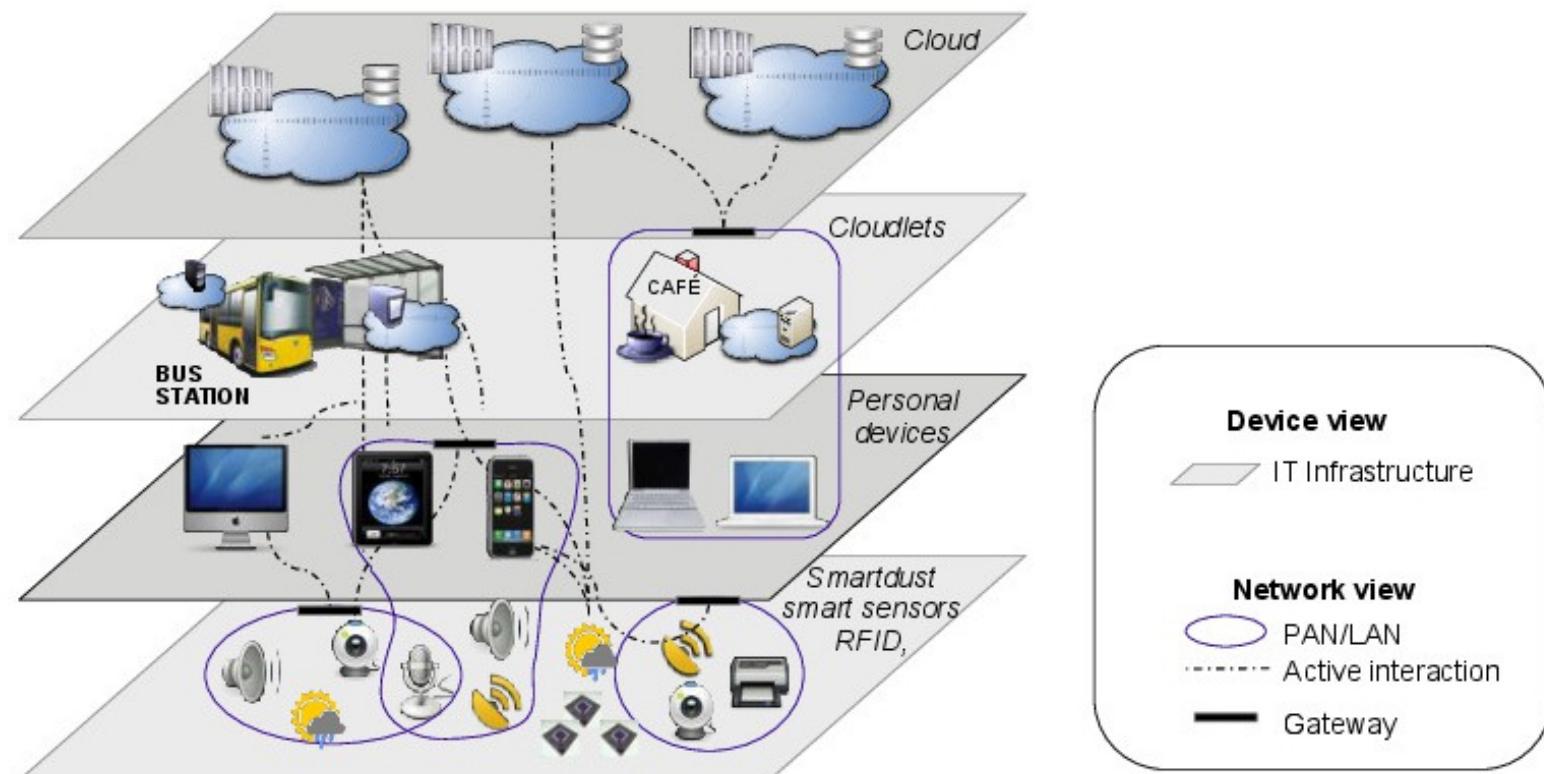
Source: GAO based on Federal Aviation Administration information. | GAO-15-221



Trafic aérien sur la France

II - Positionnement

- Problématique : chaque « type » d'infrastructure possède ses outils, ses méthodes... et si on veut un système hybride ?
 - Vision en couche (échelles)



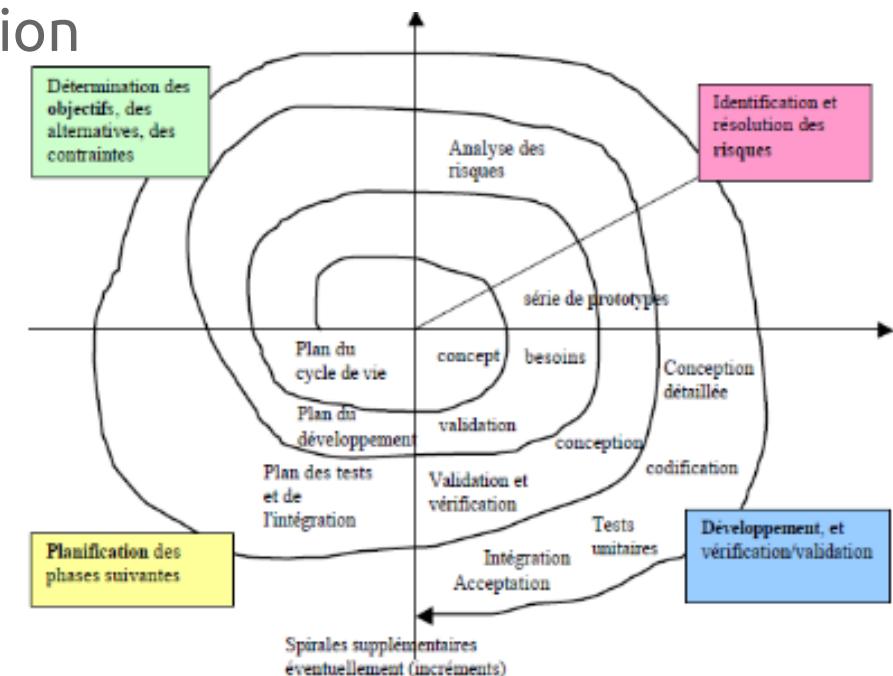
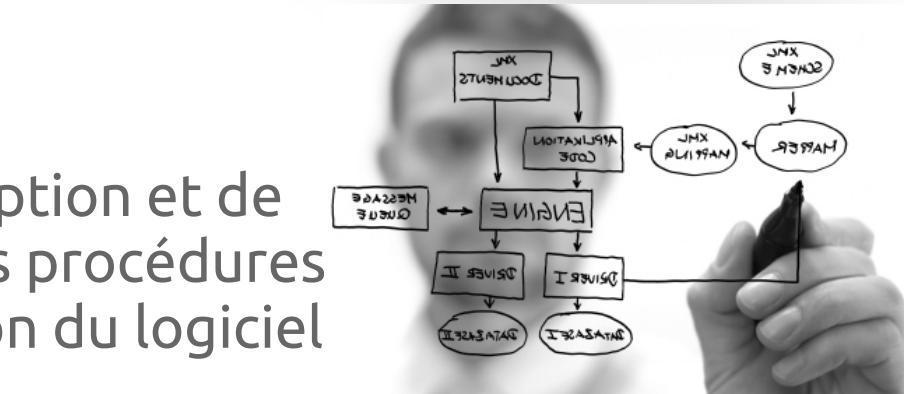
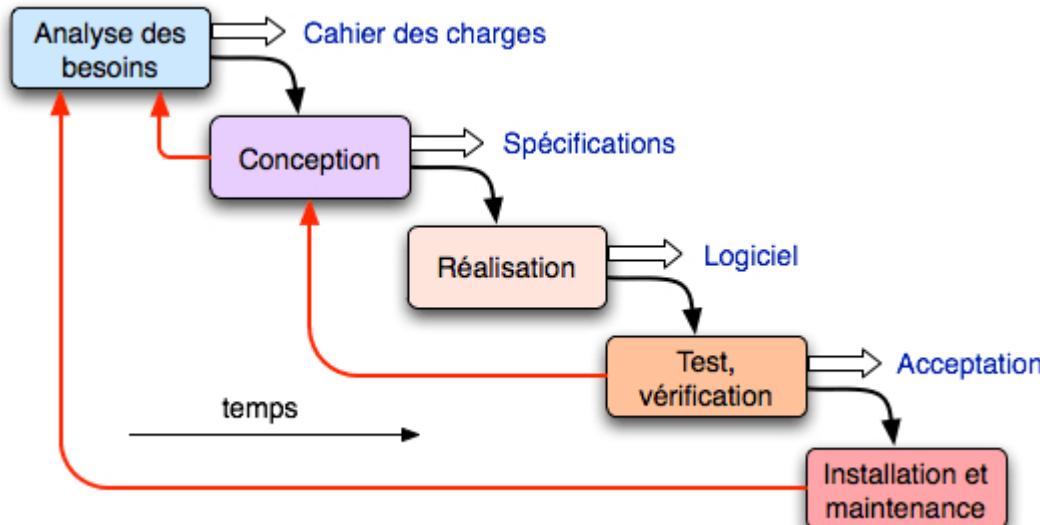
II - Positionnement



- Principales caractéristiques des systèmes répartis complexes
 - Grand nombre d'entités
 - $10^3 - 10^9$ et +
 - Forte hétérogénéité technique
 - Système, réseau, architecture...
 - Architecture dynamique (volatilité)
 - Imprévisible à la conception
 - Différents niveaux d'interactions entre entités
 - Suivant les couches de l'infrastructure
 - Volume des données échangées
 - « Big data »

II - Positionnement

- Approche : génie logiciel ?
 - « ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi »
 - « méta niveau » de la programmation



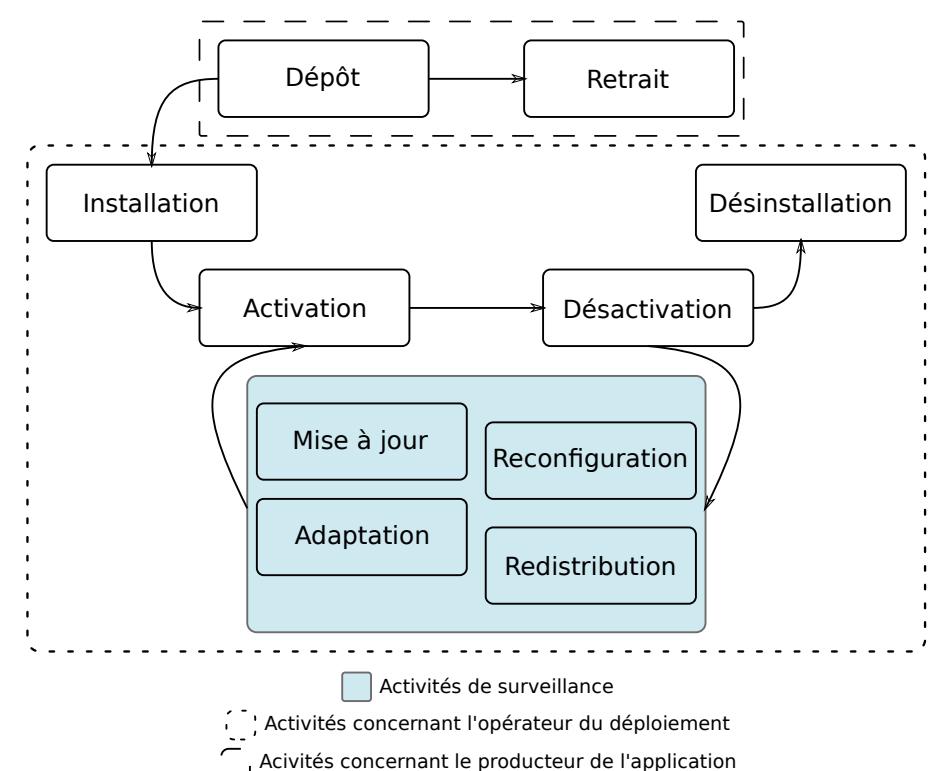
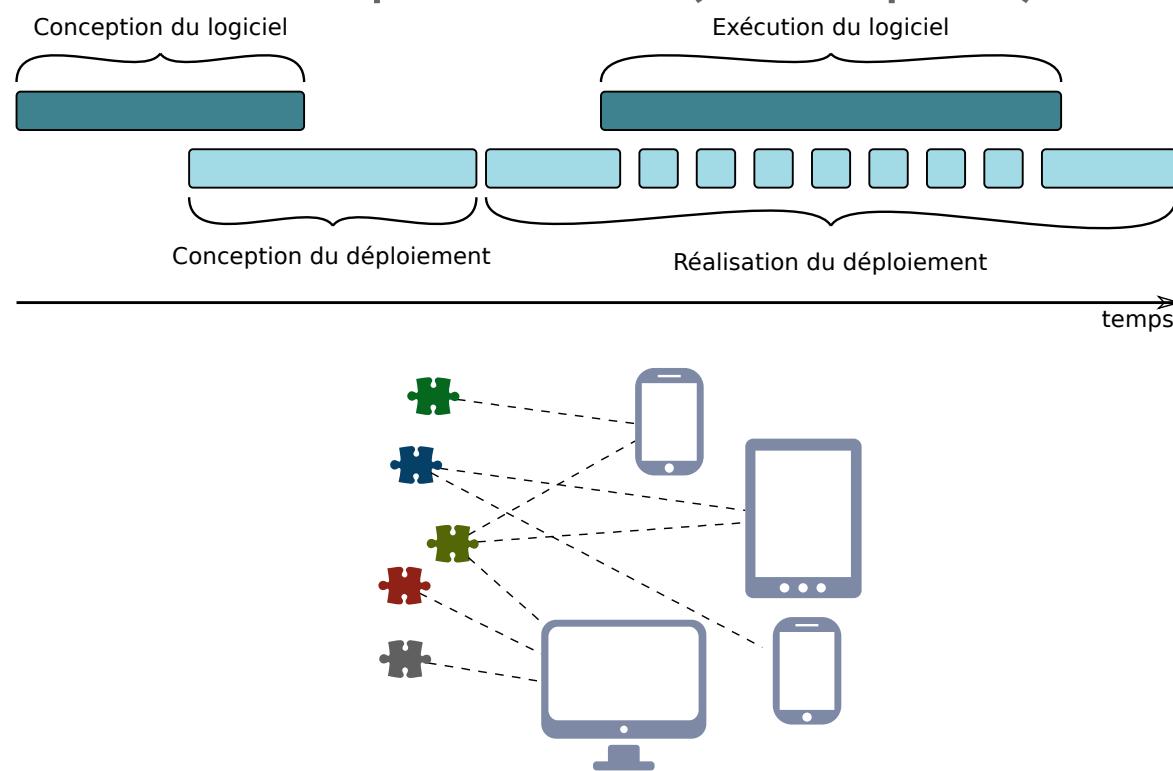
Besoins au niveau conception



- Pour interagir efficacement avec ces capteurs et effecteurs, on a besoin de « recettes » (au sens *design pattern* par exemple), de méthodes, d'outils dédiés au niveau programmation
- Intergiciels (API, abstraction, couche intermédiaire)
- Utilisation de paradigmes différents
 - Programmation évènementielle
 - Communications asynchrones...
- Utilisation des concepts de programmation avancés
 - Threads, IHM, Drag'n Drop, ...

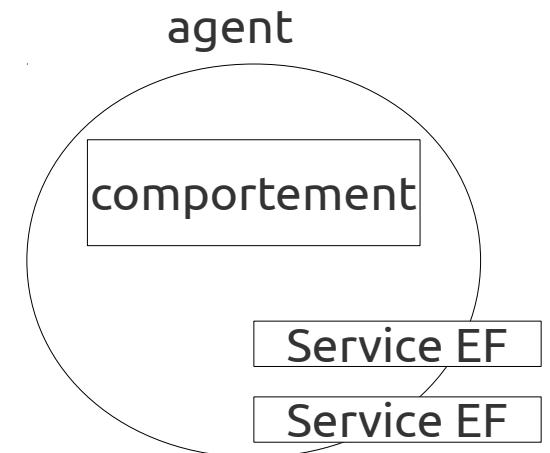
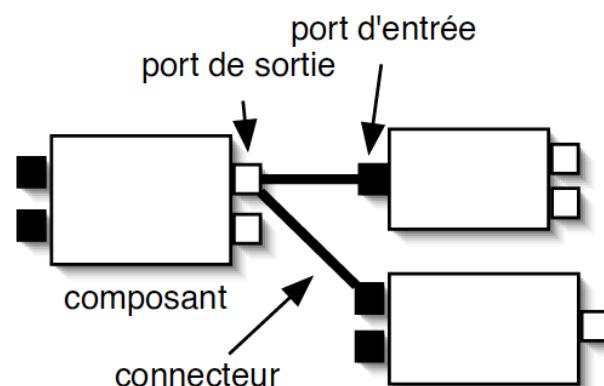
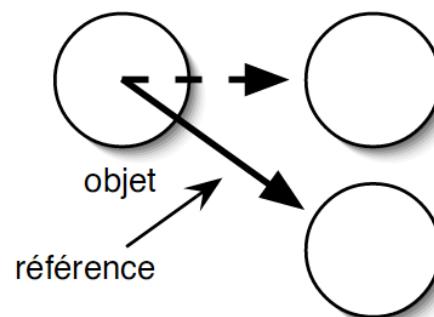
II - Positionnement

- Focus sur deux phases du cycle de vie du logiciel
 - Conception
 - Déploiement (mode push)



II - Positionnement

- Objet, Composant, Agent
 - Structuration des applications
 - Couplage / réutilisation
 - Autonomie
 - Communication asynchrone (ex : modèle d'acteur)



Résumé : des systèmes répartis complexes autonomiques – résilients ?



Inventer le ciel de demain

- Hétérogénéité
 - Capteurs, smartphones, MID, PDA, PC...
- Répartition à grande échelle
- Ouverture
 - Dynamicité, volatilité
- Complexité
 - Développement
 - Déploiement

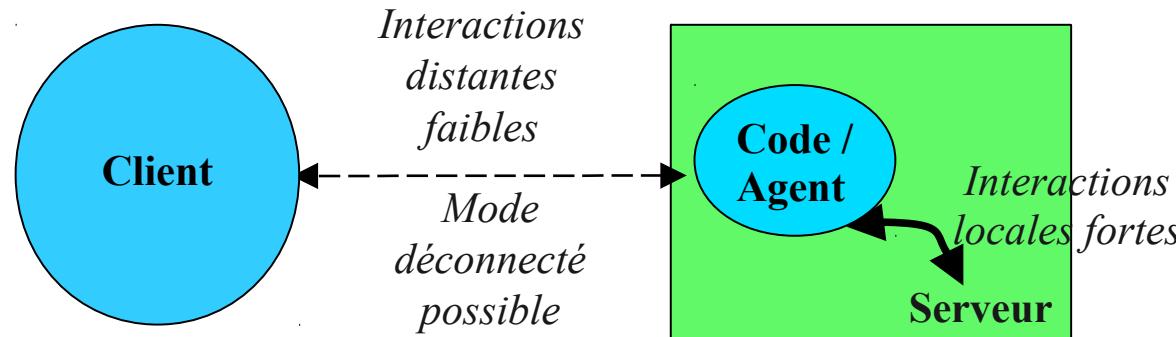


► Utilisation d'intergiciels (middlewares) adaptés

Principe du « code mobile »



- Remplacement des interactions distantes par des interactions locales



- Modèle(s) alternatif(s) au modèle C/S
- Réduction du trafic sur le réseau
- Déport de la charge

Agent Logiciel (1)



- Un *agent logiciel* est
 - une entité **autonome**
 - capable de **communiquer**
 - par messages
 - disposant de **connaissances et d'un comportement privés**
 - savoir et savoir-faire (*code*)
 - le comportement décrit comment l'agent réagit aux messages qu'il reçoit
 - intégrité
 - ainsi que d'une **capacité d'exécution propre**

Agent Logiciel (2)

- Un agent agit par délégation pour le compte d'un tiers sans être obligatoirement connecté à lui.
- Agent = outil de conception
 - unité de structuration des applications
 - support de décentralisation
 - concurrence
 - activité et évolutions sont codées au sein de l'agent lui-même
- cf. Les *agents intelligents* et les *systèmes multi-agents* de l'IAD

Agents mobiles (1)



- Un *agent mobile* est un agent logiciel qui peut se déplacer de manière autonome sur un réseau, en transportant à la fois
 - son code
 - ses données
 - son état d'exécution
- Exécution en séquence sur plusieurs sites
 - arrêt de l'exécution (sur le site d'origine)
 - transfert
 - reprise de l'exécution après déplacement (sur le site)

Agents mobiles (2) - Vocabulaire



- Une *place* est une machine physique ou virtuelle qui héberge des agents et où ils s'exécutent (serveur d'agent)
 - Granularité plus fine que le site (physique)
- Un *domaine* est un ensemble de places reliées par un réseau de communication
- Un *itinéraire* est une suite de places qui décrit un chemin dans le réseau.
 - l'itinéraire n'est pas forcément prédéfini mais peut l'être

Agents mobiles (3) - Apports



... en termes d'exécution, quand la ressource réseau est prépondérante

- Réduction du temps d'exécution de l'application
 - en limitant les interactions coûteuses (sur le réseau) en volume ou en nombre
- Limitation de la consommation de la ressource réseau
 - au bénéfice de l'ensemble du système
- Opérations déconnectées du client

Agents mobiles (4) - Apports



*... en termes d'adaptation et de déploiement,
quand le système est instable, que l'on ne peut
pas pré-installer l'application*

- Déploiement dynamique
 - selon les besoins de l'application et les conditions d'exécution
 - déplacement ou communication selon les cas
- Réactivité et adaptation
 - à l'état du support d'exécution (disponibilité des ressources)
 - à l'état du traitement en cours (résultats obtenus)

Agents mobiles (4) – Apports



... sur le plan « génie logiciel » mais passage de message (et code mobile) suffisent au cas par cas. Cependant ...

- **Le modèle d'agent mobile offre un cadre unificateur (paradigme) pour le développement et le déploiement d'applications ouvertes réparties à grande échelle (dont ubiquitaire)**
 - modèle alternatif ou extension du modèle C/S ?
 - abstraction
 - prise en compte et contrôle par le programmeur de la localité des composants des interactions
 - la répartition n'est pas cachée mais explicite

Peu d'applications réelles, pas de « killer application », pourquoi ?

- Pas d'application qui ne puisse pas être développée sans utiliser les agents mobiles
 - l'argument « cadre unificateur » est insuffisant
 - poids de l'existant
- Problèmes
 - sécurité
 - mise en œuvre
 - contrôle des agents, politiques de déplacement
- Diffusion et maturité de la technologie, réalité des besoins

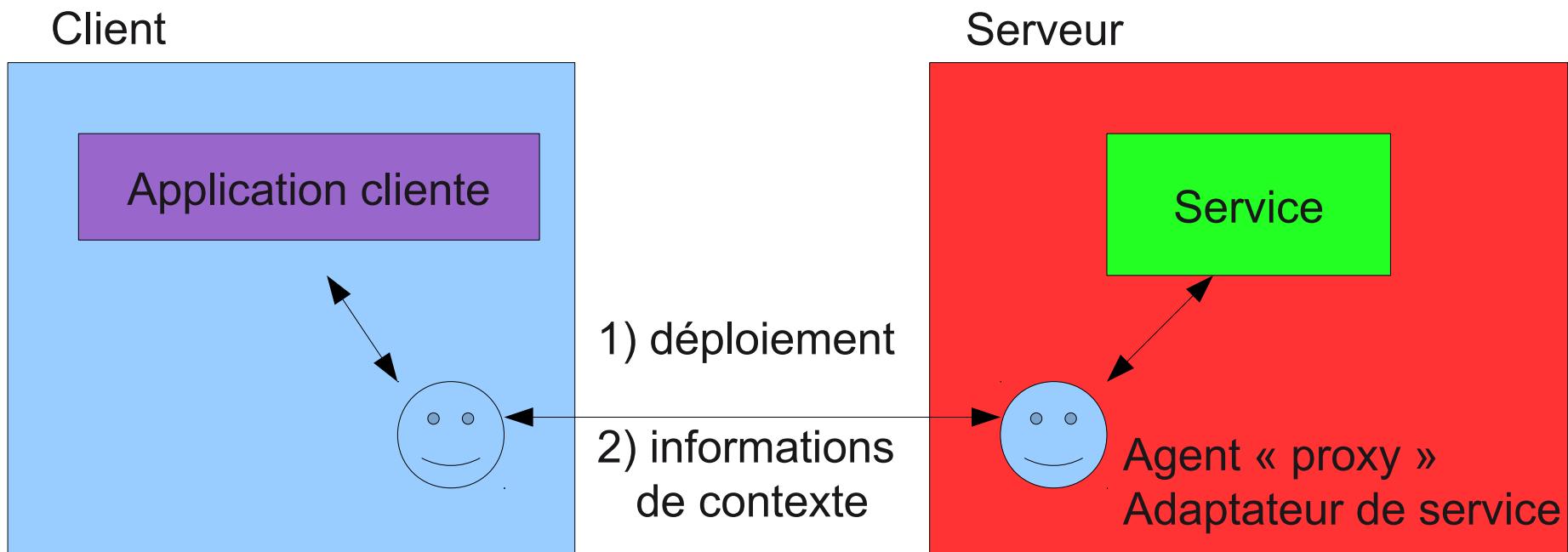
Adaptation dynamique de service



Inventer le ciel de demain

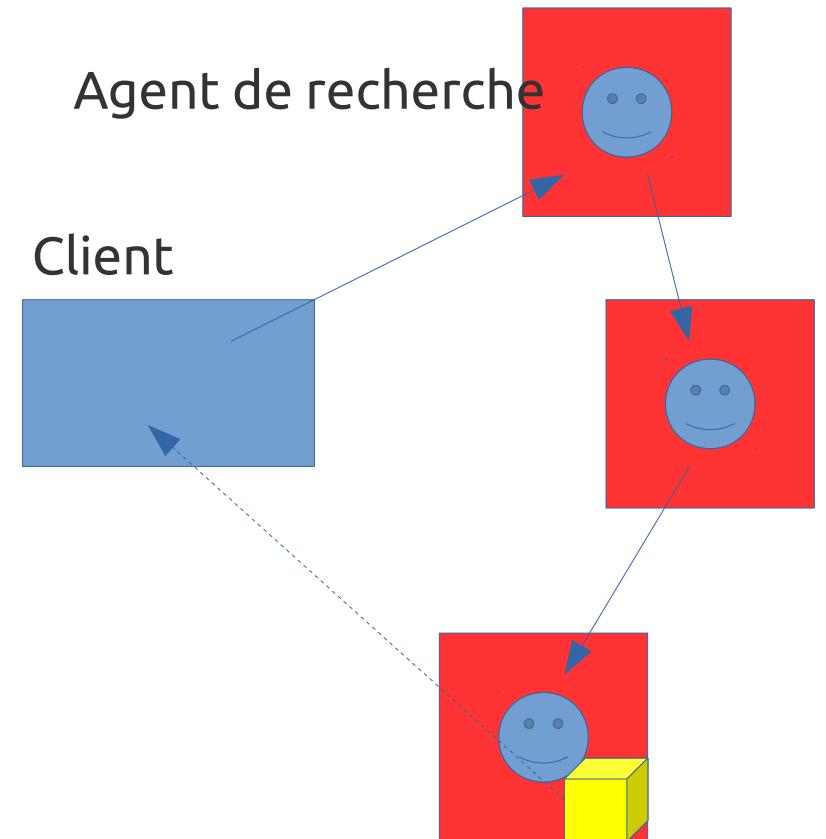
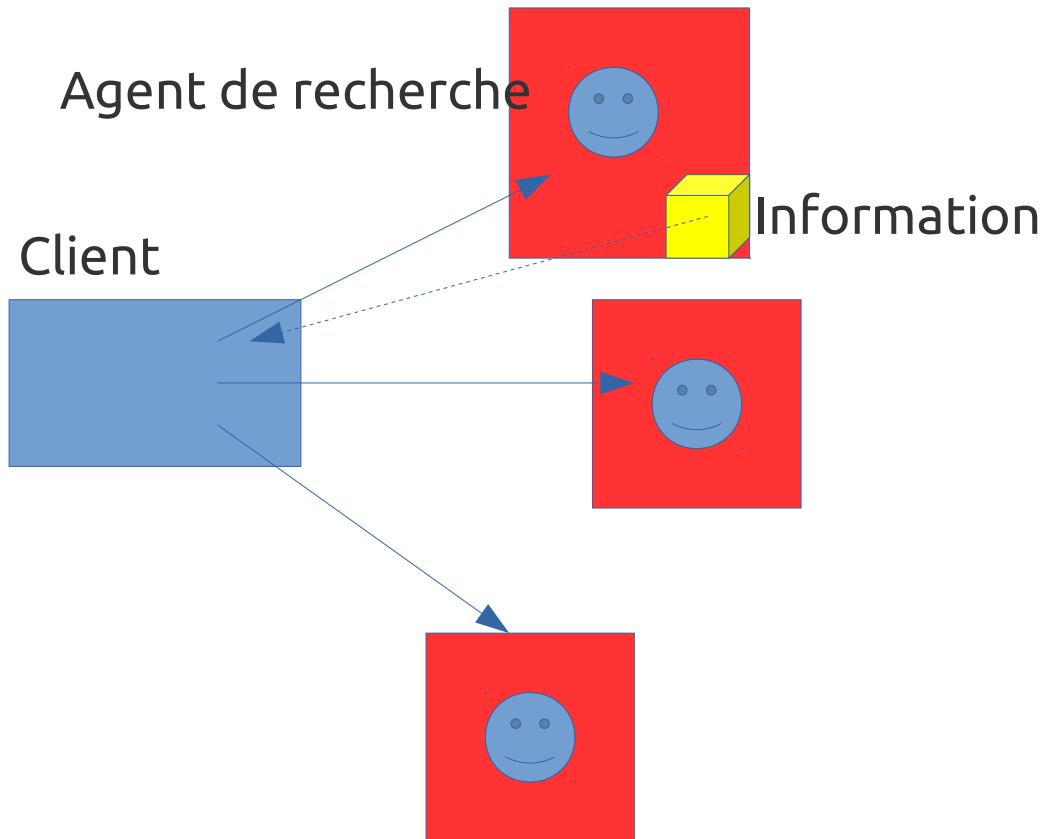


- Adaptation de vidéo à la bande passante client
- Traduction (changement de protocole)
- Filtrage contextuel
- Support de la gestion des déconnexions



Data mining

- Recherche d'information
 - Parallèle ou par itinéraire (éventuellement dynamique)

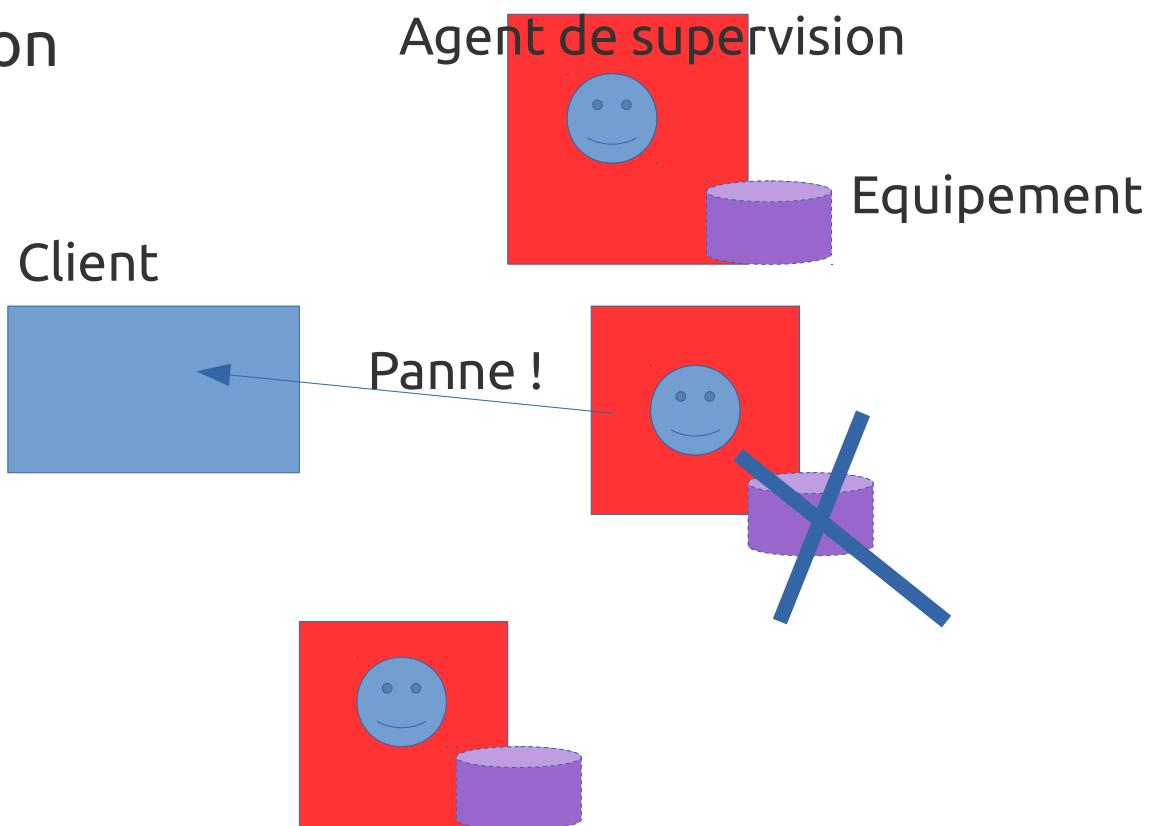


Supervision d'équipements répartis



Inventer le ciel de demain

- Remontée d'alarmes
- Détection de pannes (collaborative)
- Détection d'intrusion
- Auto-réparation
- Mise à jour

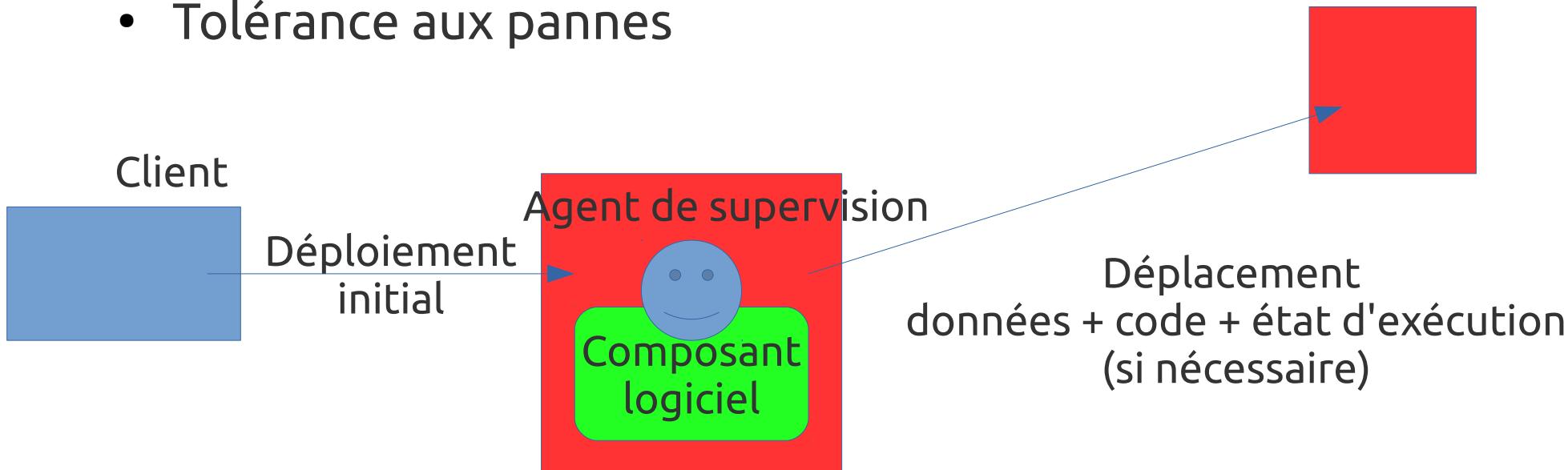


Déploiement et supervision de logiciel



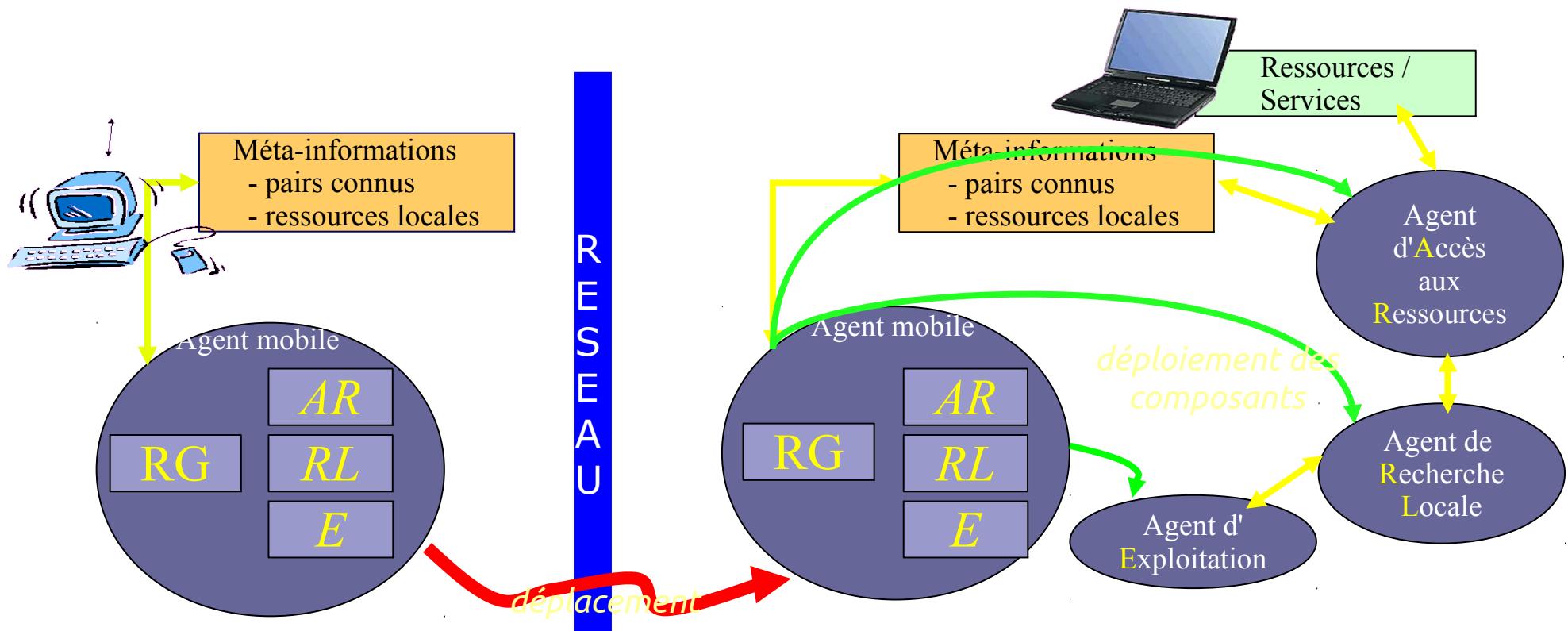
Inventer le ciel de demain

- Déploiement / installation de patch de sécurité
- Déploiement de composants logiciels
- Supervision de composants logiciels
- Adaptation dynamique à la charge
- Tolérance aux pannes



Axes de recherche

- Design patterns à base d'agents et de composants



Axes de recherche



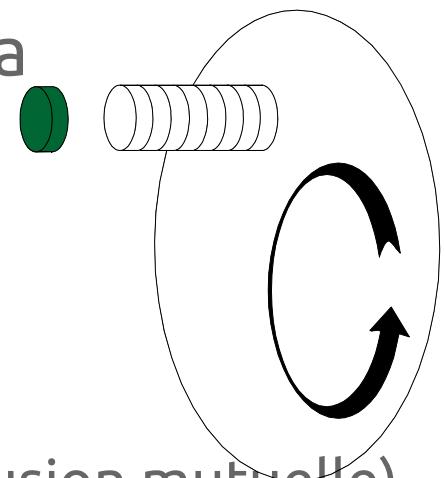
- Déploiement autonome
 - Reprendre les concepts de FDF/Deployware
 - Ok sur les grilles, mais pas pour les systèmes pervasifs
 - Utiliser les agents mobiles
 - Autonomie (contrôle décentralisé)
 - Déploiement (encapsulation+mobilité)
 - Adaptation (services non-fonctionnels)
 - Re-déploiement ?

JavAct : intergiciel à agents mobiles (1)



Inventer le ciel de demain

- Un modèle d'agent élémentaire
 - Le modèle d'acteur de Hewitt & Agha
 - Identification par référence
 - Comportement privé
 - Communication par message
 - Point à point, oneway, asynchrone
 - Traitement des messages en série (exclusion mutuelle)
 - SEND
 - CREATE
 - BECOME



JavAct : intergiciel à agents mobiles (2)



- Développée à l'IRIT (Equipe MAY)
 - Distribuée sous licence LGPL (open source)
 - <http://javact.org>
- Java standard
 - JSDK 1.4 (JRE 1.3)
 - java.rmi (Security Manager, Garbage Collecting, etc.)
 - java.Thread (caché au programmeur)
- JVM standard, pas de prétraitement de code
 - Un petit générateur de code (plugin Eclipse)

JavAct : intergiciel à agents mobiles (3)



Inventer le ciel de demain

- Agent = Objet serveur en attente de message
 - Localisé sur une machine virtuelle, mobile
 - Accessible par envoi de message (localisation transparente)
- Comportement d'acteur = Objet Java
 - Privé, encapsulé dans l'acteur (pas d'appel de méthode direct)
 - Implante une interface
 - Invocation via un envoi de message à l'acteur
 - Asynchrone (non bloquant pour l'émetteur)
 - Méthodes void : pas de retour de résultat
 - Méthodes non void : retour de résultat différé (futur)
 - Message = Objet Java « sérialisable »

JavAct : intergiciel à agents mobiles (4)



Inventer le ciel de demain

- Comment développer une application ?
 - Définition
 - Des interfaces des comportements
 - Pour chaque acteur, de l'automate des comportements
 - Génération de code
 - Classes message
 - Classes abstraites pour les comportements
 - Codage
 - Des classes comportement