

# Le TEMPS (réel) dans les télécommunications

FTR&D DTL/ASR ,DTL/TAL

P .C om bes ,Journées systèmes embarqués ,

17 Juin 2003

Le présent document contient des informations qui sont la propriété de France Télécom . L'acceptation de ce document par son destinataire implique, de la part de ce dernier, la reconnaissance du caractère confidentiel de son contenu et l'engagement de ne faire aucune reproduction, aucune transmission à des tiers, aucune divulgation et aucune utilisation commerciale sans l'accord préalable écrit de France Télécom R & D

( diffusion  
contrôlée )

# Plan



- ❖ Le domaine des télécommunications
- ❖ Systèmes en fouie et contraintes temporelles
- ❖ Enjeux et Défis
- ❖ Quelques axes de recherche (RNRT, IST)
- ❖ Perspectives

# Le domaine des télécoms.



- ➔ Convergence des télécoms, de l'informatique et du multimédia
- ➔ Développement de l'intelligence dans les réseaux et les terminaux
- ➔ Une architecture *extrêmement* complexe
  - ✓ Distribuée, largement connectée
  - ✓ Hétérogène
    - Prise en compte de l'existant
    - Hétérogénéité des technologies, des modèles d'exécution, des terminaux utilisateurs
  - ✓ De multiples standards, et en constante évolution
  - ✓ Un grand nombre d'acteurs
  - ✓ Contraintes fortes
    - Par l'environnement: utilisateur, env. radio
    - Par les PF: embarquées, cœur de réseau
- ➔ Une réelle scalabilité: des millions d'utilisateurs sur la même architecture, sice n'est sur la même application

(diffusion  
contrôlée)

# Le domaine des télécoms.



## ➔ Systèmes dynamiques

- ✓ Des applications/composants peuvent être déployés, détruits, reconfigurés à tout instant
  - Dans un environnement en exécution (pas de re-boot du système)
- ✓ (re)Configuration (de réseau) avec flexibilité en matière de choix de débit, de classes de services, de garanties de QoS
- ✓ Configuration d'applications pour assurer la meilleure qualité de service (en particulier dans l'architecture mobile)

## ➔ Architecture ouverte

- ✓ Interfaces de programmation d'application accessibles et ayant
  - un accès fin aux gestionnaires de ressources
  - et également un accès aux composants d'infrastructure avec la possibilité de les modifier ou de les remplacer

# Systemes enfouis



- ➔ Term inaux m obiles : system es non isolés
  - ✓ Fortes contraintes de l'environnem entm obile (radio) : environnem enten constante évolution , variant avec les déplacem ents de l'usager ,
  - ✓ Contraintes en tem e de ressource etpuissance de calcul (batteries) ,
  - ✓ Contraintes de tem ps critiques ,
  - ✓ Le respectdes contraintes de Q oS des tem inaux devra être en grande partie géré par les serveurs etautes plate-form es du réseau
    - ✓ G estion dynam ique des applications , négociation en tem e de Q oS
    - ✓ Recherche du m eilleur com prom is charge du tem inal/débitetcoût des com m unications
- Les *sphères* propres à un utilisateur (dom icile , distractions , travail) doivent rester simultaném entaccessibles en tout lieu avec une garantie contractuelle de bon fonctionnem ent
  - M aîtrise du partage de ressources
  - Sécurité etabsence d'interaction négative

( diffusion  
contrôlée )

# Systemes enfouis



- Mobilité des terminaux et utilisateurs MAIS AUSSI du code
- ➔ PF de cœur de réseau, serveurs d'application
- ➔ Réseau et communication
- ➔ Objets communicants : domaine avec des utilisations « routage pair à pair » pour former des pico-réseaux « spontanés »
- ➔ Un système embarqué est lui-même constitué de plusieurs sous-systèmes embarqués, répartis et communicants
- ➔ Systèmes enfouis : systèmes qui doivent rester *inconnus* des utilisateurs finaux
  - ✓ Utilisés pour améliorer l'interface usager et non l'inverse
  - ✓ Boîte noire qui doit respecter (sur ses interfaces) des contraintes fonctionnelles et non fonctionnelles
  - Interfaces formellement spécifiées et comportement vérifié

(diffusion  
contrôlée)

# Contraintes temporelles



- ➔ À tous les niveaux de l'architecture et des applications
  - ✓ Protocoles et composants de gestion de ressources, de contrôle, etc
  - ✓ Couche de médiation et communication
  - ✓ Composants de service et application finale
- ➔ Du Temps réel dur au temps réel mou
  - ✓ Dur : dépasser les contraintes même de peu n'a pas d'intérêt (protocole)
  - ✓ Mou : si possible ne pas dépasser mais minimiser le dépassement est intéressant (applications « end users » : streaming video/audio, ...)
- ➔ Aspects qualitatifs et quantitatifs (probabilité, moyenne)
  - Performance et maîtrise de l'utilisation des ressources
- ➔ Hétérogénéité des modèles d'exécution :
  - ✓ Synchrone et asynchrone
  - ✓ Séquentiel d'entrelacement, atomisation des actions

(diffusion  
contrôlée)

# Enjeux



## ➔ Enjeux:

- ✓ Continuité d'accès/ubiquité, environnement virtuel
  - ✓ Connectivité sans couture (besoin)
  - ✓ Recherche de localisation (utilisateur, service): les *plus proches* et disponibles
- ✓ Intégration avec l'existant, *prévoir* le futur
- ✓ Création, déploiement et personnalisation rapide de nouveaux services
- ✓ Réduction et maîtrise des coûts,
- ✓ Ouverture, concurrence: → respect des contrats
  - ✓ La QoS comme atout d'une offre concurrentielle

## ➔ Evolution des besoins dans le monde des télécommunications

- ✓ Enjeux d'abord de nature économique (relation coût/performance)
- ➔ Besoins de sûreté de fonctionnement de plus en plus durs:
  - ➔ sécurité, confidentialité, responsabilité par contrats dans un univers ouvert pour les composants et services,

(diffusion  
contrôlée)



- Maîtriser l'hétérogénéité
  - des composants et des mécanismes de communication,
  - de l'environnement (acteurs, environnement radio, etc)
  - des techniques logicielles,
  - des modèles d'exécution (synchrone/asynchrone)
  - ✓ Obtenir une transparence pour chaque utilisateur/acteur
  - ✓ Offrir une architecture et un chaîne de conception horizontale
  - ✓ APIs formellement définies, intégrant les contraintes de temps et des procédures de négociation en terme de ressources
  - Actuellement: Utilisation des PF existantes fondées sur les normes *de fait*: profil JAVA
  - Perspectives: Action sur les PF futures pour obtenir des PF ouvertes et adaptables à des modes de gestion des ressources (ie. du temps de calcul) conforme aux exigences QoS.

- Assurer la sûreté de fonctionnement (avec contraintes de Temps et de performance) de systèmes complexes
  - ✓ Complexité en terme de nombre de composants en fouis et en terme des fonctionnalités offertes par ces composants (procédures d'adaptation, de négociation)
  - ✓ Systèmes ouverts == systèmes de *confiance*
  - Nécessité de modèles très expressifs
  - Vers la vérification positionnelle
- Evaluer les rapport coût/performance en amont (avant le déploiement)
  - ✓ Taxation précise et contractuelle, avec les clients, entre opérateurs, etc
  - ✓ Partager une plate\_forme (des moyens de communications) entre applications dans le meilleur rapport coût/qualité
- ✓ Des besoins souvent contradictoires

# Vérification de code efficace



- Garantir les temps de réponse à l'exécution d'une application
- Application type « protocole radio » (Temps réels dur)
  - ✓ Programmation Synchrones (compilation vers code exécutable)
  - ✓ Modélisation du comportement temporel à partir du code machine (et non à partir du code source ce qui serait plus classique mais moins fidèle !) sous forme d'Automates Temporisés
  - ✓ Spécifications des contraintes temporelles
  - ✓ Validation sur modèle par *simulation exhaustive*
  - ✓ (=> Projet RNRT TAXYS (VERMAG FT R&D))
- Cette approche est non limitée à une programmation synchrone
  - ✓ peut être reprise dans le cas de PF nominalisée (ie. Java) pourvu que celles-ci soient **ouvertes** : composants fins de gestion de ressources accessibles et modifiables.
- ✓ Synchrones (ESTEREL) choisis dans cet exemple pour sa bonne adaptation aux applications à fortes contraintes temps réels dur

(diffusion  
contrôlée)



# Techniques de preuves

- Application aux composants critiques
- ➔ Vérification de la conformité d'un protocole complexe
- ➔ Ex: équipement de contrôle de conformité ATM (capacité ABR )
  - ✓ famille de fonctions du temps paramétrée par l'instant courant
  
- ➔ RNRT/CALIFE : environnement logiciel permettant de construire de façon cohérente des modèles de systèmes temporisés
  - ✓ Modélisation du système par p-automates
  - ✓ Abstractions permettant d'obtenir un modèle restreint pour outils spécialisés
  - ✓ Environnement graphique de spécification (sorties vers Coq, Kronos, HyTech, etc)
  - ✓ Résultats
    - ✓ Théoriques (décidabilité, logique + réécriture + modules)
    - ✓ Améliorations d'outils de preuve
    - ✓ Applications : modèles de protocoles multicast (PGM, PIM-SM)

(diffusion  
contrôlée)

# Modélisation SDL et vérification



- ➔ IST/INTERVAL, approche SDL:
  - ✓ Intégration des techniques dans des environnements de génie logiciel standards
  - ✓ Application à des systèmes de taille importante
  - ✓ Simulation (prototypage) et vérification exhaustive
- ➔ Une Application temps réel complexe : Protocole (Reliable Multicast Transport Protocol II)
  - ✓ Multicast sur IP (tous types de réseau : ADSL, satellite, 100 Mb/s ... )
  - ✓ Contrôle de la vitesse d'émission : le temps de transmission est un paramètre du système d'adaptation
- ✓ Résultat concluant malgré la complexité du problème
  - ✓ La vérification exhaustive arrive à ses limites.
  - ✓ La simulation est possible en restreignant les valeurs de certains paramètres
  - ✓ La simulation est utile pour permettre une meilleure compréhension du système
    - ✓ Observation de comportements parfois surprenant
    - ✓ Possibilité d'obtenir une trace d'exécution et de la rejouer pour comprendre les mécanismes mis en œuvre.

(diffusion  
contrôlée)

# UML: services et composants



- ➔ IST/Omega : Développement correct de systèmes embarqués temps réel avec UML
- ➔ Application FTR & D : Composants de service et infrastructure pour la composition
  
- ➔ Objectif:
  - ✓ Construire rapidement une application (distribuée) fiable répondant à des besoins avec contraintes de temps
  - ✓ Vérifier l'intégration de cette application avec d'autres applications déployées sur la même infrastructure
    - ✓ Comportement système (vue utilisateur) de la composition de différentes applications,
  - ✓ Construire des composants d'infrastructure (de service) fiables et réutilisables (donc vérifiées et composables)
  - ✓ Services et composants de service avec contraintes de temps : mobilité, disponibilité, localisation,
- ➔ Temps réel et modèle de composants pour UML

# Prédiction de performances



## ✓ Constat:

- 80% de logiciels du type « Client/Serveur » sont à refaire à cause de problèmes de performances
- Peu d'études de performance en phase de conception
  - Outils de performance difficiles d'accès pour les concepteurs d'architecture de systèmes logiciels
  - Deux domaines différents, fonctionnelle et performance, deux modélisations et domaines d'expertise différents

## ➤ Objectifs de la prédiction des performances:

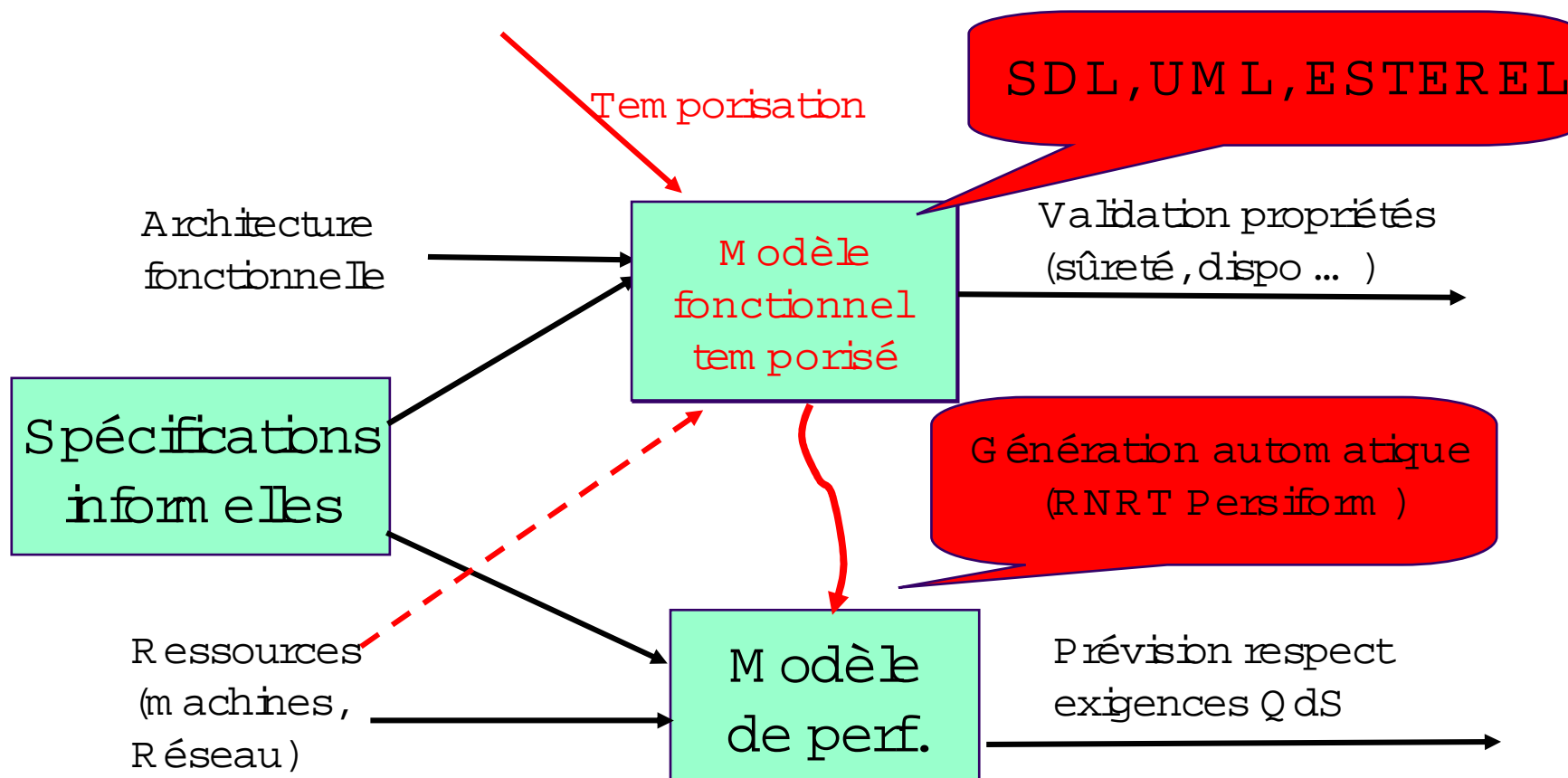
- Réduire les coûts, maîtriser les ressources, mutualiser les plateformes
- Construire une architecture en fonction de critères de performance et coût
- Analyser les impacts des ressources sur les performances de composants

## ➤ Intégration dans les cycles de développement

- Approche cohérente entre analyse fonctionnelle et analyse des performances
- Intégration avec des environnements industriels

(diffusion  
contrôlée)

# Démarche de modélisation



(diffusion contrôlée)

# Perspectives



- ➔ Objectif: Infrastructure logicielle (middleware) et chaîne de conception intégrées pour le développement et la mise en œuvre d'applications avec garanties de QoS temps réel.
- ➔ Besoins en terme de techniques logicielles
  - ✓ Expression de propriétés sur le temps et les performances des composants et systèmes (langages *clairs* et expressifs)
  - ✓ Modélisation et outils pour:
    - La vérification formelle : techniques de preuve
    - La vérification semi-formelle : simulation interactive ou guidée
  - ✓ Prototypage, prédiction de performances
  - ✓ Construction et vérification de composants respectant des contraintes de temps
  - ✓ Modèle de composants intégrant les contraintes de temps, de performance et le partage des ressources
  - ✓ Production et exécution de tests en environnement distribué avec contraintes de temps et mesure des performances

(diffusion  
contrôlée)