Plate-forme de composants logiciels pour la coordination des adaptations multiples en environnement dynamique

Djalel CHEFROUR
INRIA Rennes (2001-2004)
Projet RNRT Cyberté

Plan de l'exposé

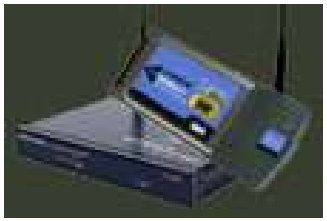
1. Introduction
   - Problème : adaptation dynamique aux variations des ressources
   - Contexte : projet RNRT Cyberté
2. Démarche suivie
   - Objectif à atteindre
   - Approche adoptée
3. Solution proposée : plate-forme ACEEL
   - Modèle de composant (Canevas) Aceel
   - Service de surveillance
   - Intergiciel de coordination des adaptations
   - Expérimentations : navigateur web, vidéo à la demande
4. Conclusion
   - Bilan et Perspectives

Plan de l'exposé

1. **Introduction**
   - Problème : adaptation dynamique aux variations des ressources
   - Contexte : projet RNRT Cyberté
2. Démarche suivie
3. Solution proposée : plate-forme ACEEL
4. Conclusion

1.1 Problème:

- Équipements mobiles (PDA, portables …)
  - Équipements hétérogènes
  - Ressources faibles : Mémoire, Batterie…
- Réseaux sans fils (WiFi, BlueTooth, GPRS …)
  - Bande passante, taux d'erreurs de transmission, latence…
- Variations dynamiques des ressources utilisées par plusieurs applications



Pb: Difficile d'exécuter les applications traditionnelles qui ne s'attendent pas aux fluctuations des ressources (ex: Multimédia)

R: Applications adaptables

Les applications adaptables

- Adaptation : modifier un logiciel en fonction de son environnement pour avoir une QoS acceptable
  - Réactive **vs** évolutive
  - Dynamique **vs** statique

- Niveaux concernés
  1. Utilisateur (conscient [Efstratiou02] ou non)
  2. Applications (ad hoc [Rumor, Guy99], complexe)
  3. Intergiciel (spécialisables [Ensemble,vanRene98])
  4. Système (adaptation transparente [Coda, Braam98])

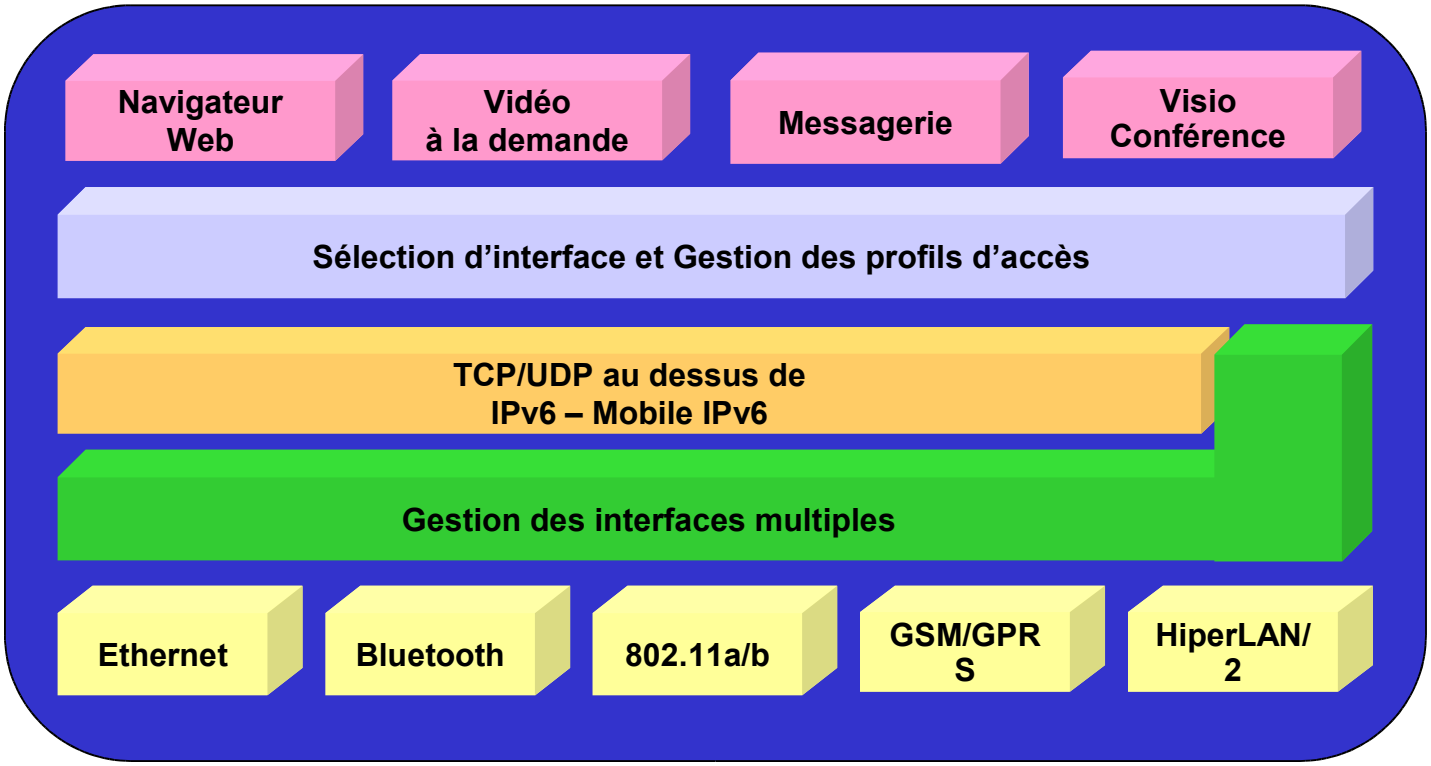
→ Applications conscientes de l'adaptation avec aide d'un canevas et d'un intergiciel pour coordonner les adaptations multiples

Historique

- Plate-forme MolèNE [M.T.Segarra 2000]
  - Composants réactifs en environnement mobile, projet Charcot (application médicale)
- Plate-forme AeDEN [F.Le Mouël 2001]
  - Service de distribution adaptative et dynamique des applications en environnement mobile
- Plate-forme ACEEL
  - Coordination des adaptations multiples en environnement mobile, projet RNRT Cyberté

1.2 Contexte : projet Cyberté

- Projet RNRT (Irisa, ENST B., U. Strasbourg, FT R&D, Cisco)
- Terminal mobile mulit-interface réseau (WiFi, GPRS, Ethernet…)



Exemple de scénario avec un terminal Cyberté

1. Utilisateur dans son bureau, participe à une visioconférence
   - Réseau Ethernet
   - Vidéo et son haute qualité

2. Utilisateur change de salle pendant la conférence
   - Réseau WiFi
   - Vidéo et son de moins bonne qualité

3. Utilisateur poursuit sa conférence dans la rue
   - Réseau GPRS
   - Pas de vidéo. Seule la voix est transmise

Plan de l'exposé

1. Introduction
2. **Démarche suivie**
   - Objectif à atteindre
   - Approche adoptée
3. Solution proposée : plate-forme ACEEL
4. Conclusion

2.1 Objectif

- Faciliter le développement d'applications adaptables, en offrant un modèle de composant et des outils d' interactions avec l'environnement.

  - Factoriser les mécanismes génériques de l'adaptation

  → maximiser la réutilisation

  - Offrir des langages expressifs pour écrire ce qui est spécifique
  - Masquer les détails des différentes plates-formes

  → simplifier la tâche du développeur

  - Considérer le cas de plusieurs applications distribuées

  → coordonner les adaptations multiples

  - Tenir compte des préférences de l'utilisateur

  → assurer une QoS acceptable

2.2 Approche adoptée (1)

- Adaptation = modification, donc plus facile si l'approche de programmation est **modulaire**.

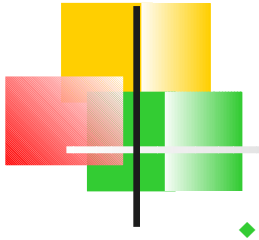
Les apports de la modularité :

5. Identifier les sujets de l'adaptation

→ Définir des types d'actions d'adaptation

6. Identifier les dépendances entre entités adaptables

→ Coordonner les adaptations multiples



2.2 Approche adoptée (2)

- Composant = interface + état + implantation(s)

  Adaptation individuelle

- Application = composants inter-agissants, distants

  Il y a des dépendances de flots

  → Problèmes d'adaptations incohérentes

- Système = applications concurrentes

  Il y a des dépendances de partage de ressources

  → Problèmes d'adaptations conflictuelles

2.2 Approche adoptée (3)

Adaptation = problème complexe

Opérer une séparation des préoccupations :

1. Détecter et notifier les variations pertinentes
2. Décider des actions coordonnées à réaliser (afin d'optimiser la QoS)
3. Exécuter les actions choisies

- Différents mécanismes :

  Distinguer les type de codes qui mettent en œuvre l'adaptation

2.2 Approche adoptée (4)

1. Le code métier
   ex : Parser HTML, Encodeur vidéo, etc.

4. Le code qui notifie les variations pertinentes
   ex : chute de la bande passante réseau

8. Le code qui spécifie quand et comment adapter un composant
   ex : événement – condition – action

11. Le code qui exécute les actions d'adaptation
    ex : modifier un paramètre, une implantation, un appel…

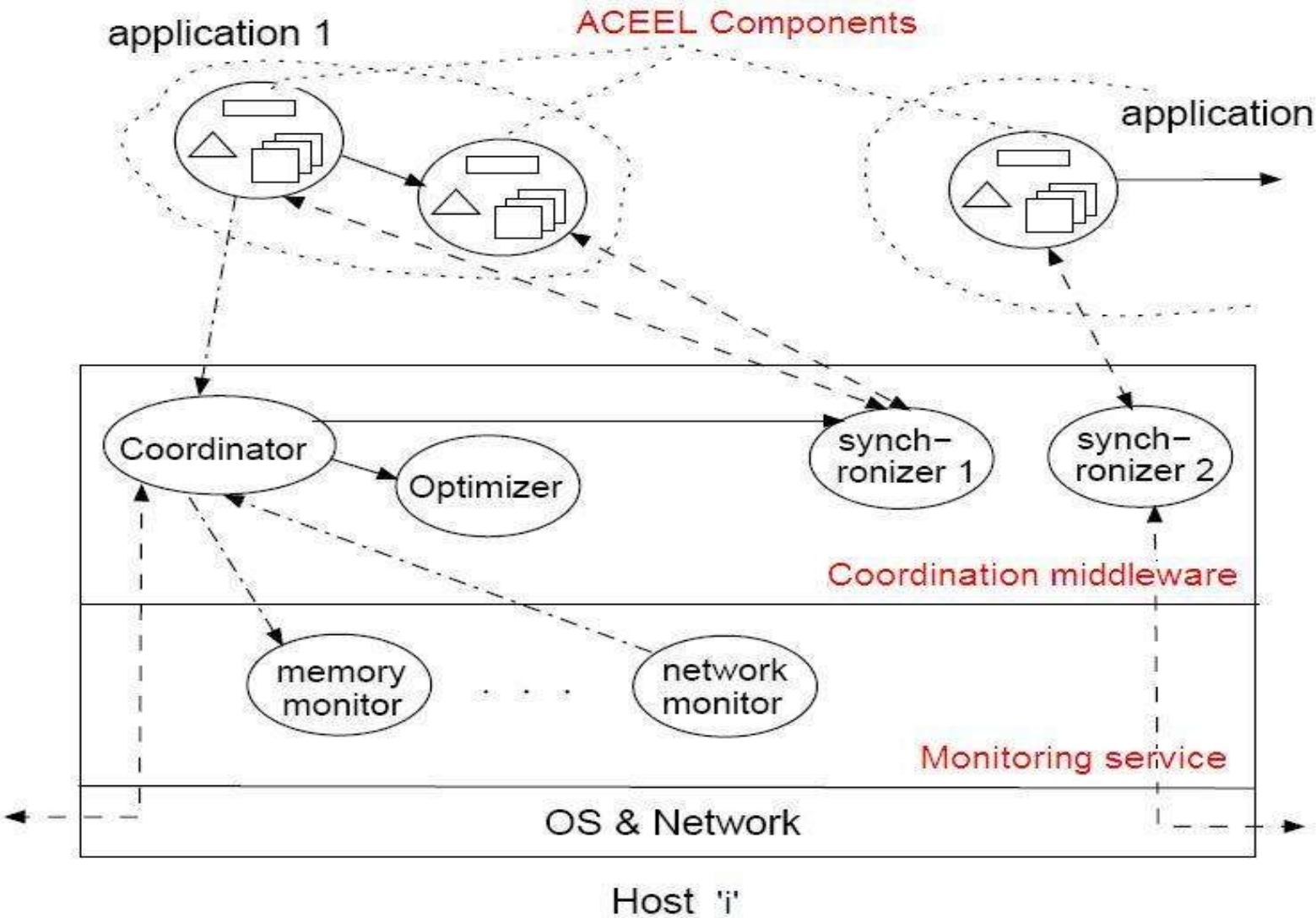
14. Le code qui spécifie quels composants s'adaptent ensemble
    ex : si l'émetteur change le format des données, il faut changer le décodeur

17. Le code qui décide des adaptations de façon coordonnée
    ex : éviter les conflits, synchroniser…

Plan de l'exposé

1. Introduction
2. Démarche suivie
3. **Solution proposée : plate-forme ACEEL**
   - Modèle de composant Aceel
   - Service de surveillance
   - Intergiciel de coordination des adaptations
   - Expérimentations : web, vidéo
4. Conclusion

Plate-forme ACEEL



3.1 Composants ACEEL : principes du modèle

- Avoir un mécanisme d'introspection de l'état
- Posséder des éléments modifiables dynamiquement
  - des implantations interchangeables, des paramètres d'état adaptables ou des appels de méthodes réifiables

Les modifications ont un impact sur :

- La consommation des ressources par le composant
- La qualité de service perceptible par l'utilisateur

→ Chaque élément adaptable au sein du composant définit une **dimension d'adaptation**

- Ex: fréquence d'un son, taille d'une image, nb de couleurs, etc.

3.1 Composants ACEEL : politique d'adaptation

- Définit les caractéristiques de ce qui est adaptable
  - État du composant : séparé des implantations ou interne
  - Implantation : atomique, interruptible ou re-démarrable
- Définit les dimensions d'adaptation

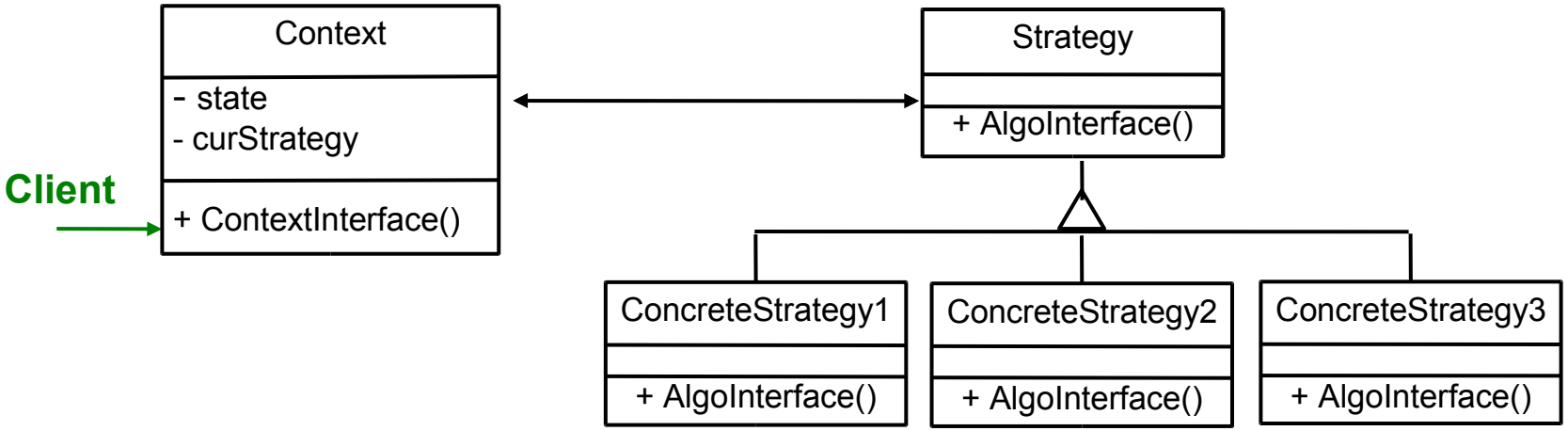
  dimension = un poids + { règles d'adaptation }

  règle d'adaptation = association entre
  - Une condition sur l'environnement d'exécution
  - Des actions d'adaptation qui ont un coût et qui engendrent un changement de la qualité de service
- Définit la classe de priorité du composant

Techniques utilisées (1) : Patron Strategy

- Patron de conception : une conception réussie qui résout un problème récurrent, catalogué par [GoF en 1995]
- Patron Strategy: utilisation de plusieurs variantes d'un algorithme

  Exemple : composant d'affichage de données (tableau, diagramme, cercle…)

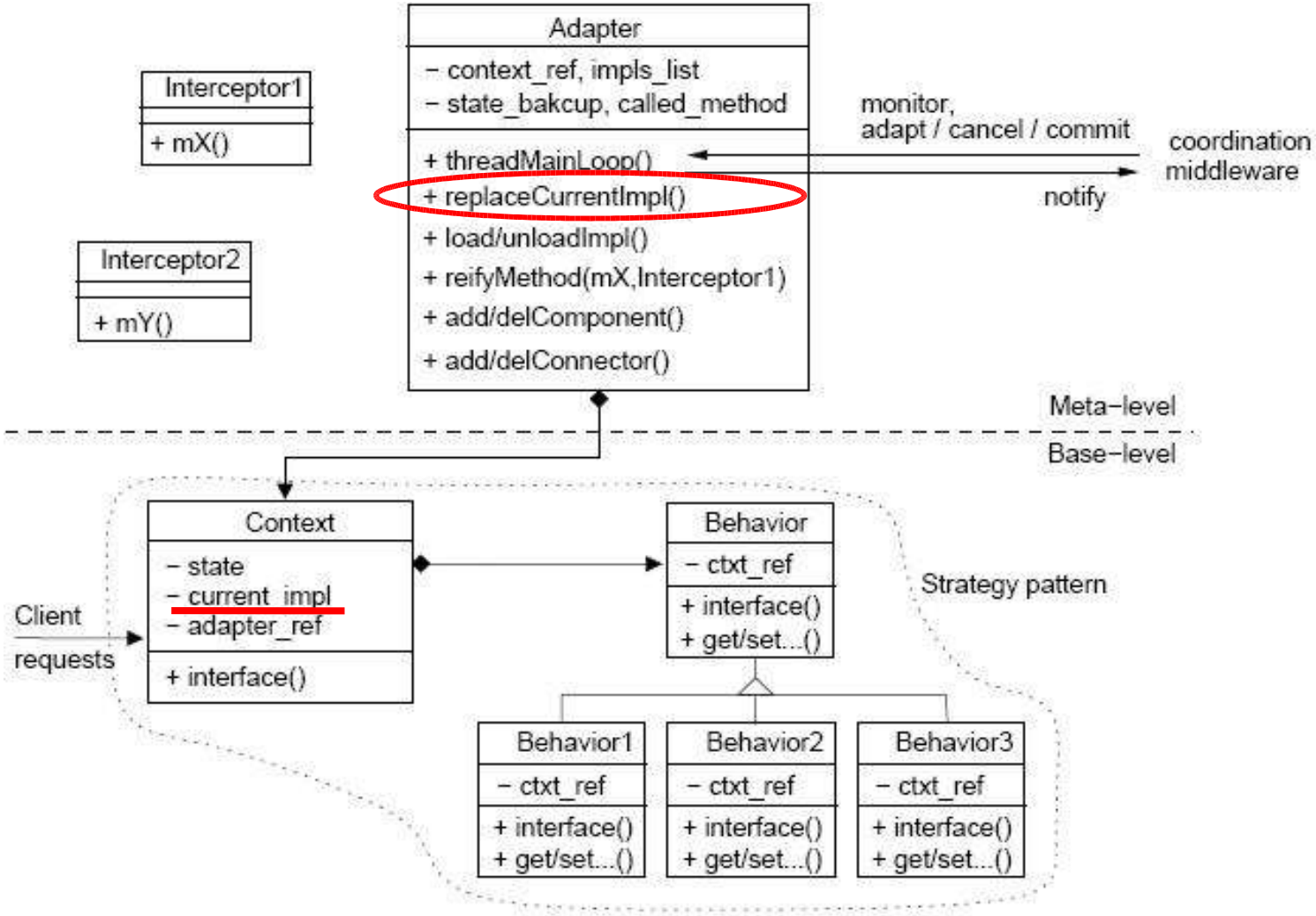


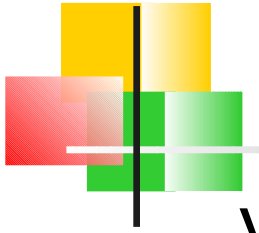
Techniques utilisées (2) : Réflexion

- Introduite par B. Smith en 1982
- Système réflexif : capable d'inspecter et de modifier sa propre interprétation de son domaine d'application (la représentation de soi)
- Niveau de base : partie (code+données) qui traite le domaine d'application
- Méta-niveau : partie qui traite la représentation de soi

→ Permet l'introspection est l'intercession du composant

Architecture d'un composant ACEEL





Exemples de composants adaptables

Vidéo à la demande adaptable

3. Serveur vidéo
   - Paramètres adaptables :
     - taux de bits, dépend de la bande passante
     - Nb trames/seconde, dépend de la CPU client

4. Receveur flux vidéo
   - Implantations interchangeables :
     - Chacune correspond à un format de codage (mjpeg, mpeg4), dépend des erreurs de transmissions

Rôle du développeur d'un composant ACEEL

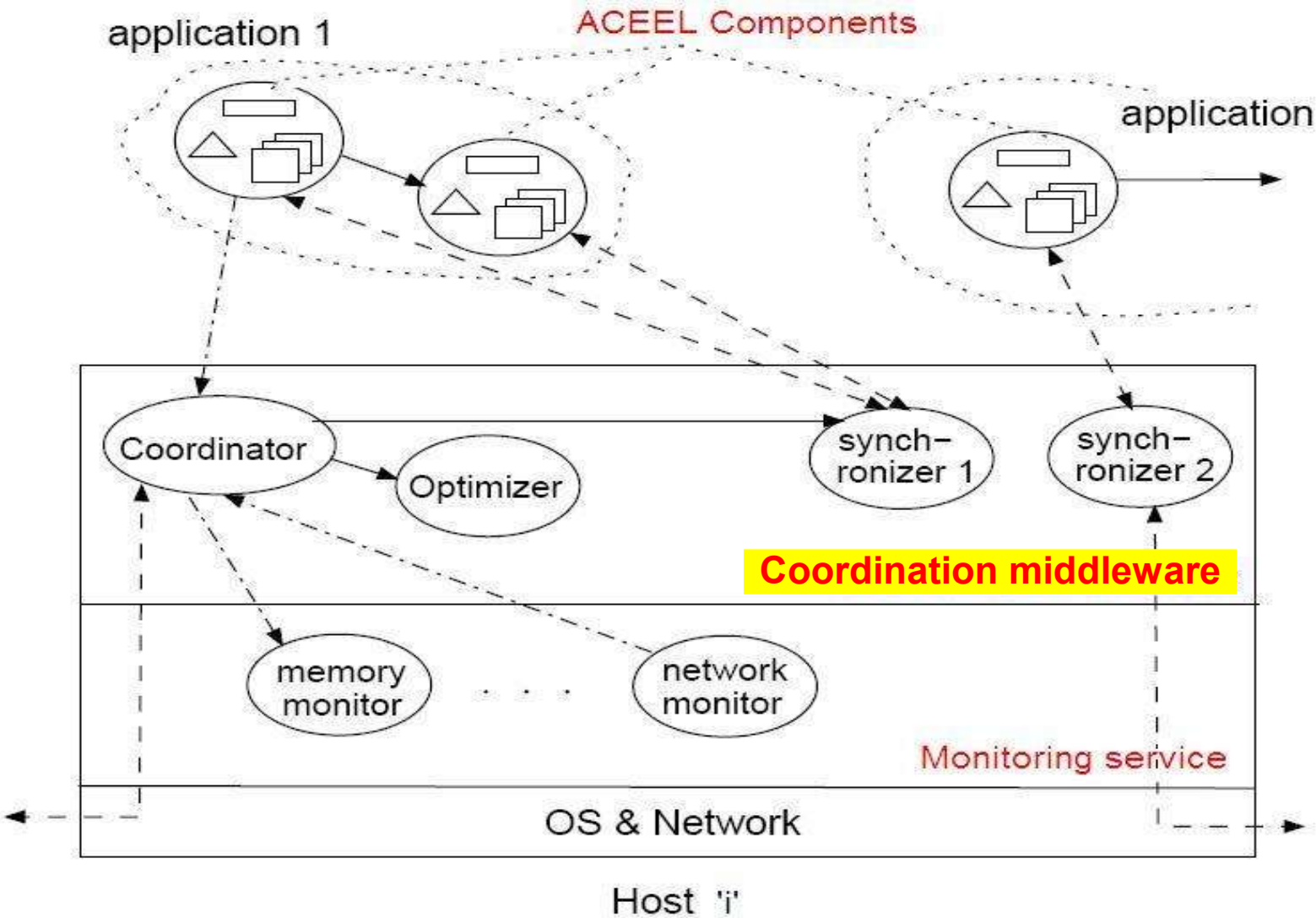
1. Écrire le code métier du composant
   - Utiliser les classes du canevas Aceel Context et Behavior, et éventuellement State et Interceptor

2. Écrire la politique d'adaptation
   - Identifier les éléments adaptables et les actions qui les modifient → dimensions d'adaptation
   - Identifier les positions possibles du composant dans les dimensions d'adaptation → indice de qualité de service
   - Estimer les ressources dans chaque configuration → conditions des règles d'adaptation
   - Affecter des poids différents aux dimensions qui dépendent, en partie, des mêmes ressources

Langage des politiques d'adaptation

```
<adaptation_dimension name="download_images" weight="1">
  <polling_periods>
    <poll resource="net[http_proxy].bandwidth" every="30s"/>
  </polling_periods>
  <adaptation rules>
    <rule quality_index="100%">
      <when resource="net[http_proxy].bandwidth" ge="56kbps" />
      <adapt action="tuneParameter" params="load_imgs=1" cost="1"/>
    </rule>
    <rule quality_index="50%">
      <any_condition/>
      <adapt action="tuneParameter" params="load_imgs=0" cost="1"/>
    </rule>
  </adaptation_rules>
</adaptation_dimension>
```

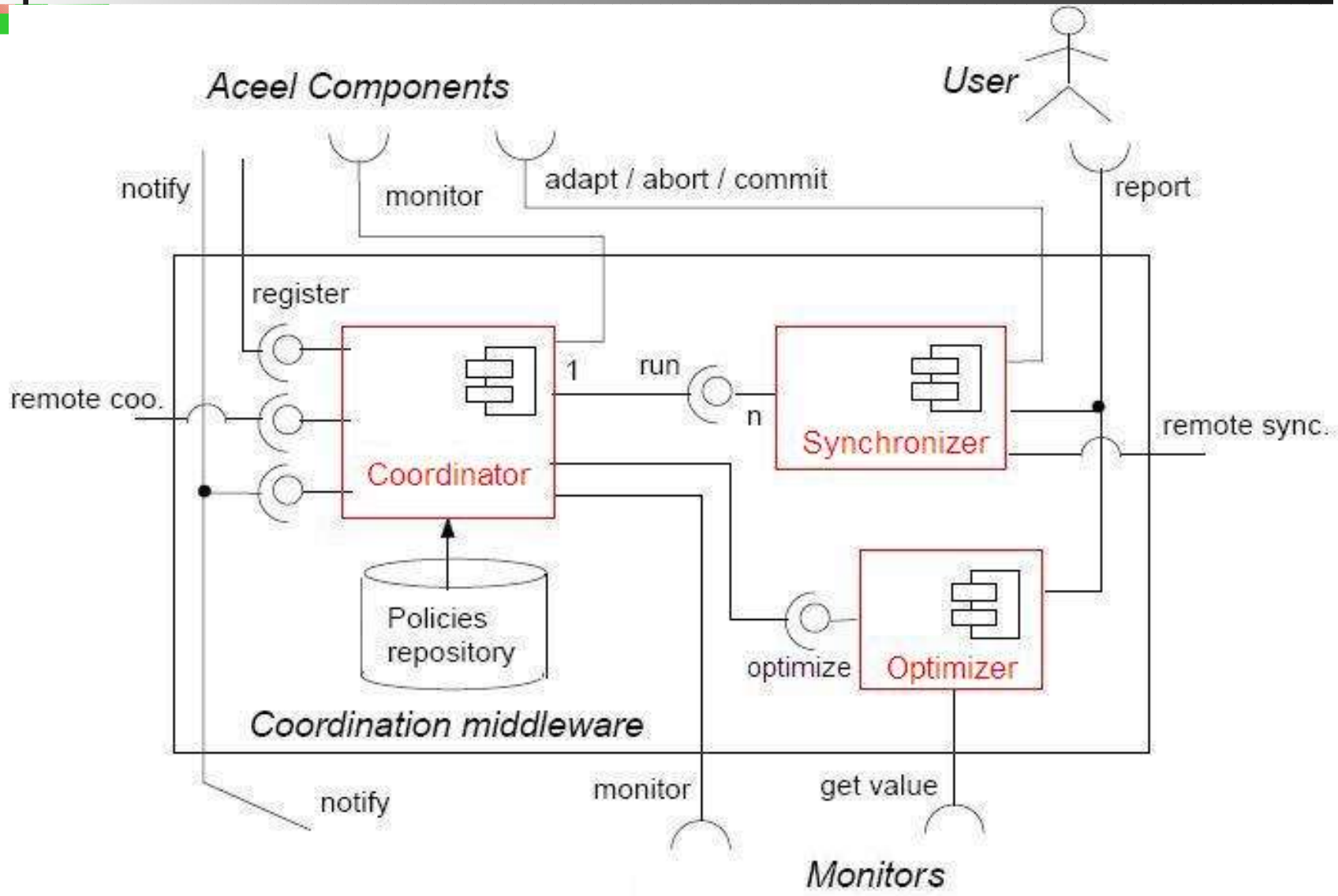
Plate-forme ACEEL



3.3 Intergiciel de Coordination des adaptations

- Prend les décisions d'adaptation des composants applicatifs de façon coordonnée
  - Enregistre les composants et interprète les politiques d'adaptation
  - Traite les événements notifiant les variations
  - Gère les dépendances entre composants inter-agissants, en utilisant une politique de coordination par application
  - Gère les dépendances de partage de ressources
  - Synchronise l'exécution des adaptations multiples
- Les composants pris individuellement et les systèmes d'exploitation ne peuvent réaliser cela

Architecture de l'intergiciel de coordination



Politiques de coordination des comp. inter-agissants

Objectif : Préserve la cohérence de l'application

- Identifie les composants adaptables d'une application et leurs sites de déploiement
- Précise les composants qui doivent s'adapter ensemble
- Spécifie comment synchroniser les actions d'adaptation de ces composants

Une politique = {règles de coordination}

Règle de coordination = {

(comp. A : action P) avec (comp. B : action R)

alors P et R parallèles, …

}

Exemple de politique de coordination

```
<coordination_rule>
  <when comp_id="Receiver" runs="replaceImpl"/>
  <synchronize>
    <sequence>
      <parallel>
        <adapt comp_id="Receiver" action="replaceImpl" params="F"/>
        <adapt comp_id="Streamer" action="tuneParamter" params="format=F"/>
      </parallel>
      <adapt comp_id="Streamer" action="tuneParameter"
          params="time_cursor=Receiver.state.time_cursor"/>
    </sequence>
  </synchronize>
</coordination_rule>
```

Rôle du coordinateur (1)

1. Interprète les politiques d'adaptation et les politiques de coordination
   1. Envoi les requêtes de surveillance pour les ressources référencées dans les règles d'adaptation
   2. Construit le Vecteur mode de fonctionnement de chaque composant local 'i'

$$X_i = [x_{i1}, x_{i2} \ldots x_{im}]$$

Exemple de l'émetteur vidéo

Xi = [BitRate, FrameRate, Format]

x_{i1} = num. règle qui spécifie le taux de bits vidéo

$$\rightarrow \quad Q(X_i) = \sum_{j=1}^{m_i} p_{ij} q(x_{ij}) \Big/ \sum_{j=1}^{m_i} p_{ij}$$

Rôle du coordinateur (2)

1. Gère les ‘n’ composants Aceel déployés sur le même site
   - $X = [X_1, X_2 \ldots X_n]$
   - $Q(X) = \sum_{i=1}^{n} c_i Q(X_i)$
   - c_i : classe de priorité du composant ‘i’ (de son application)

3. Gère les événements notifiant des variations
   1. Évalue des conditions des règles d’adaptation
   2. Lance l’optimiseur pour recalculer le vecteur X qui maximise la qualité de service
   3. Délègue l’exécution des adaptations aux synchroniseurs

Rôle de l'optimiseur

Trouver le nouveau vecteur Modes de fonctionnement (X') qui maximise la QoS sous les contraintes :

1. Disponibilité des ressources

   Seuils définis dans les conditions des règles d'adaptation

2. Absence de conflits de partage de ressource

   Somme des parts allouées aux composants partageant une ressource < seuil disponible

4. Cohérence des composants inter-agissants

   Chaque règle de coordination limite les configurations possibles pour les composants qu'elle lie

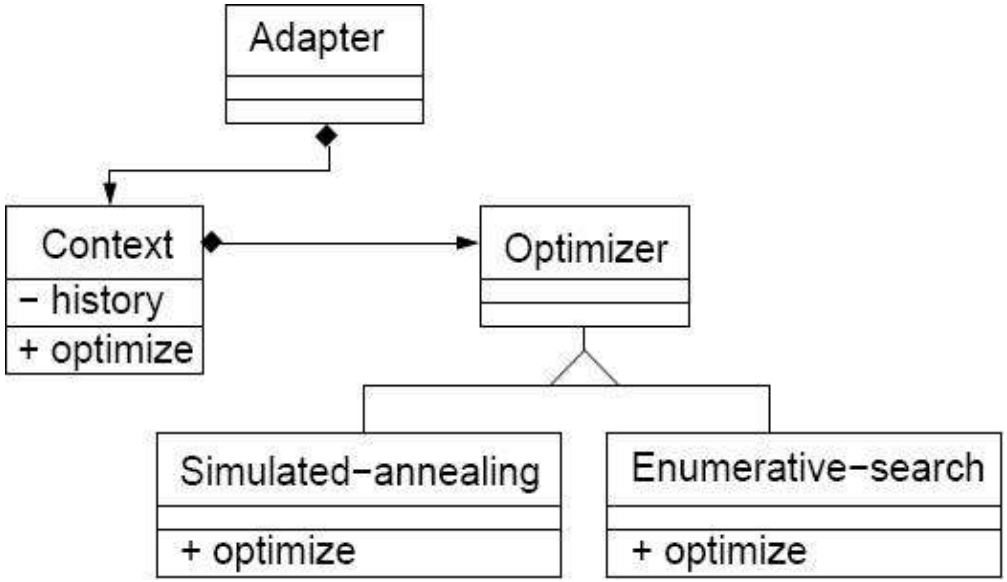
→ Problème d'optimisation distribuée sous contraintes (PODC)

Architecture de l'optimiseur

- Les PODC appartiennent à la classe NP-difficile
- Cas d'un problème centralisé
  - Résolution par un algorithme exact

    Risque d'explosion combinatoire
  - Résolution par méta heuristique (ex: recuit simulé)

    Pas de garantie de trouver la solution optimale



Cas d'applications distribuées

- Maximiser la QoS globale sur N sites

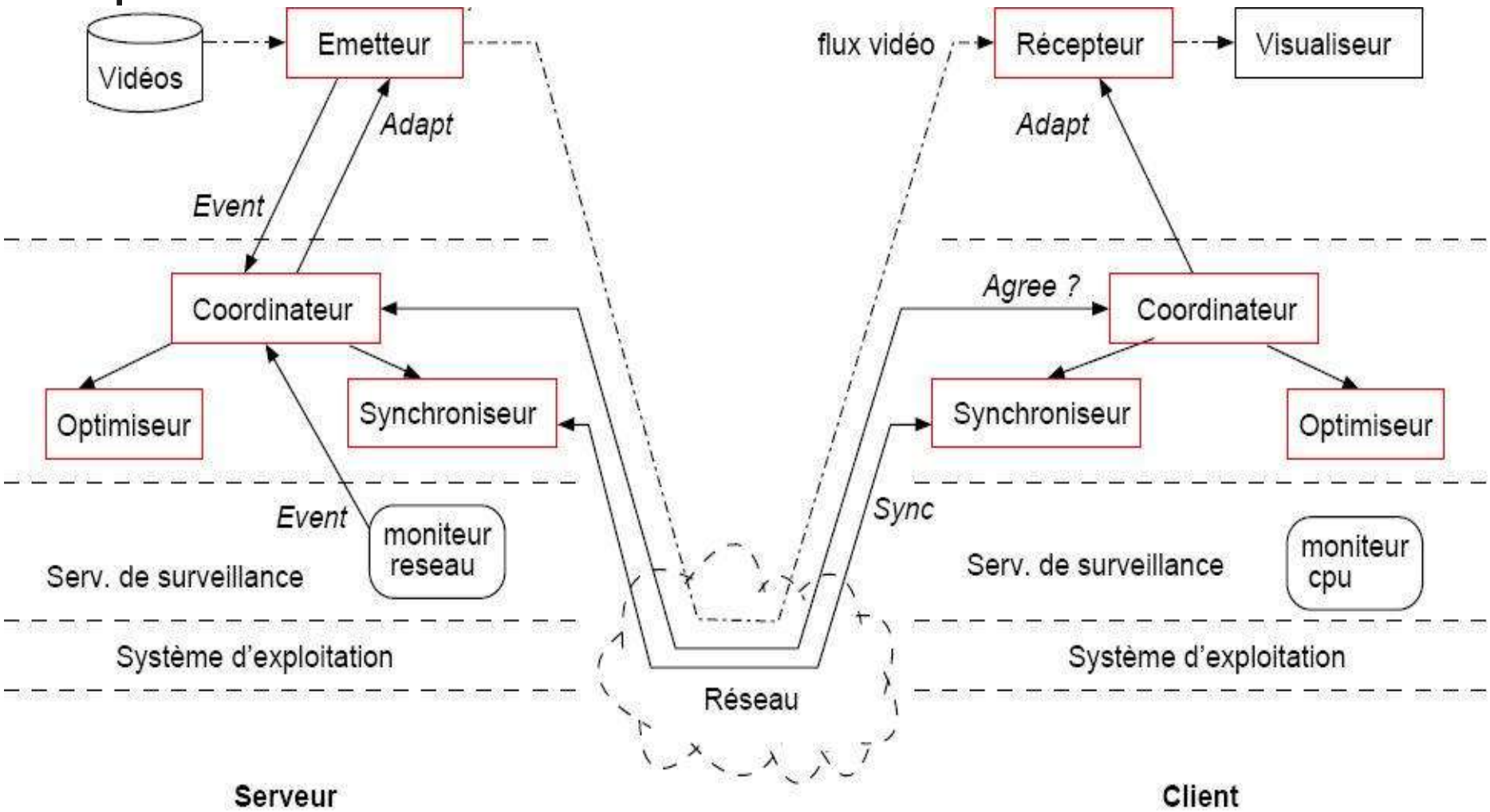
$$Qg = \sum_{i=1}^{N} Q(X^i)$$

- Utiliser un protocole d'accord distribué pour les variables de décisions soumises aux contraintes distribuées
  1. Le coordinateur de chaque site lance son optimiseur
  2. Chaque optimiseur calcul les 'k' meilleurs vecteurs « modes de fonctionnements » des composants locaux et les 'k' QoS associées
  3. Les coordinateurs des sites s'échangent les résultats de leurs calculs
  4. Chaque coordinateur peut alors trouver la QoS globale

Rôle du synchroniseur

- Il a un synchroniseur par application et par site
  - Cela permet d'isoler les adaptations des applications concurrentes
- Contrôle l'exécution des adaptations décidées pour les composants inter-agissants
  - Respecte l'ordre défini dans les règles de coordination
- Implante un protocole de commit à deux phases
  - Envoie les ordres d'adaptations aux composants concernés
  - Attend les réponses (succès ou échec) des composants
  - Envoie les ordres de confirmation ou d'annulation des adaptations

3.4 Expérimentation : vidéo à la demande adaptable



Plan de l'exposé

1. Introduction
2. Démarche suivie
3. Solution proposée : plate-forme ACEEL
4. Conclusion
   - Bilan
   - Perspectives

4. Conclusion : bilan

- Développer des mécanismes génériques pour faciliter la construction d'applications adaptables
  - Un modèle de composant réactif
  - Un service de surveillance
  - Un intergiciel de coordination gérant la cohérence et de l'absence de conflits
  - Deux langages déclaratifs et extensibles pour définir les politiques d'adaptation et de coordination
  - Intégration de la plate-forme dans un terminal Cyberté et expérimentations

4. Conclusion : perspectives

- Offrir des implantations alternatives pour le coordinateur, l'optimiseur et le synchroniseur
- Évaluer les performances du prototype
- Valider le passage à l'échelle de l'intergiciel de coordination

- Résoudre le problème d'instabilité (adaptations en chaîne ou en boucle)
- Vérifier automatiquement la couverture des règles d'adaptation
- Développer des modèles décrivant les variations de certaines ressources pour anticiper les adaptations

Extra slides

Plate-forme de composants ACEEL

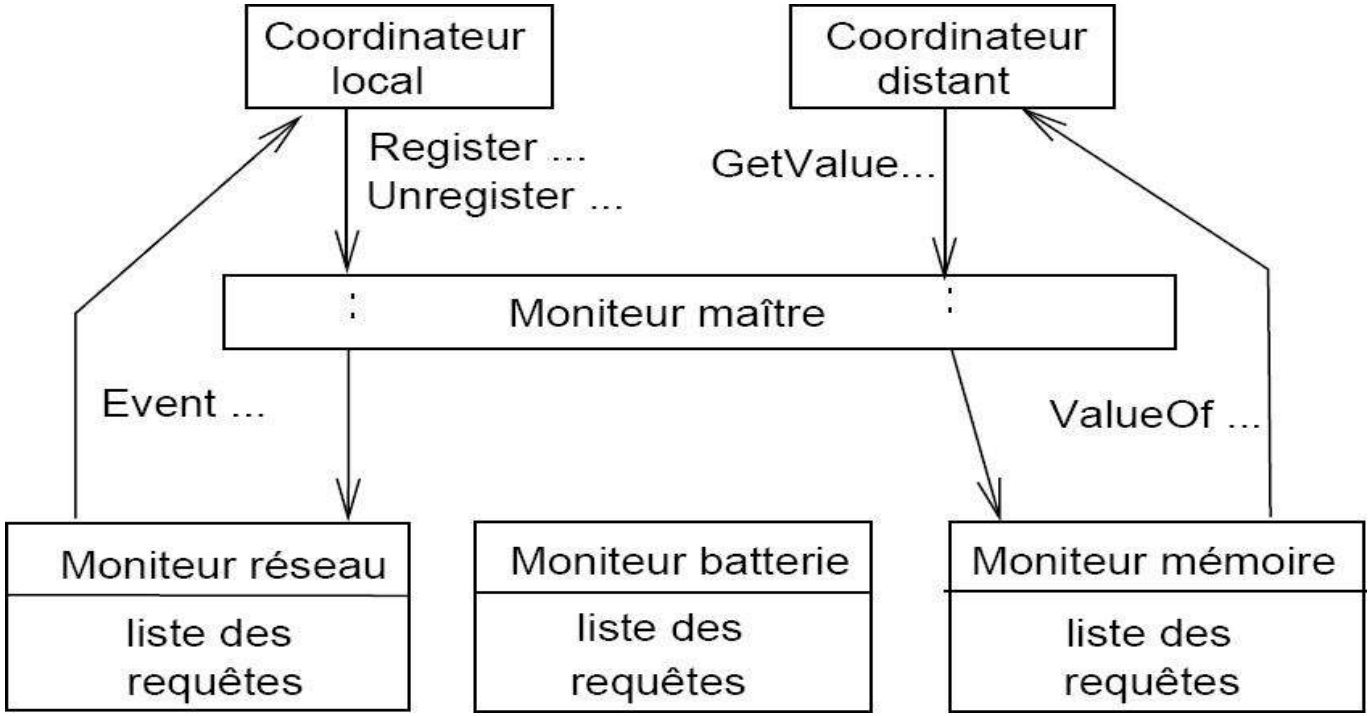
Djalel CHEFROUR
INRIA Rennes, 2001-2004 (plein-temps)
2005 (temps partiel)

Processus d’adaptation

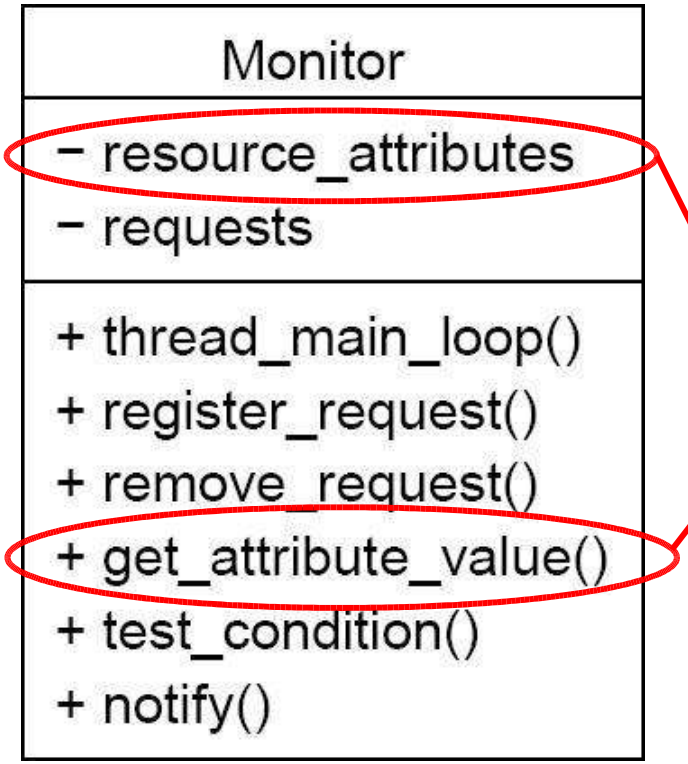
1. Geler l'interface du composant
2. Sauvegarder les infos nécessaires à une éventuelle annulation de l’adaptation
3. Exécuter les actions d'adaptation spécifiées par appel des méthodes de Adapter
4. Notifier le succès ou l'échec de cette exécution à l’intergiciel de coordination
5. Attendre l'ordre de valider ou d'annuler les actions d'adaptation réalisées
6. Dégeler l'interface du composant et reprendre le traitement des nouveaux appels clients

Service de surveillance de l'environnement

- Notification d'événements simples (seuil d'une ressource)
- Protocole au dessus de TCP



Moniteur générique



A spécialiser en fonction de la ressource et de l'API du système.