Cache de composants pour la gestion de déconnexion

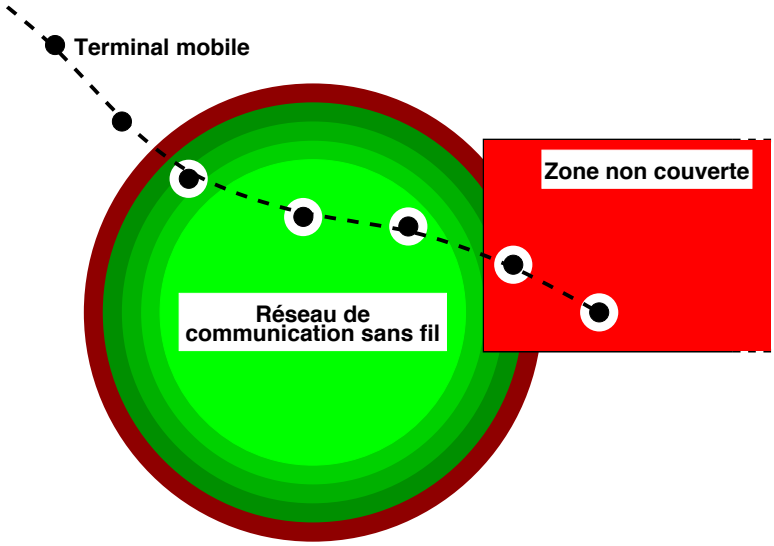
Denis Conan et Nabil Kouici

GDRI3/GTMOB, CNAM, Paris

13/01/2005

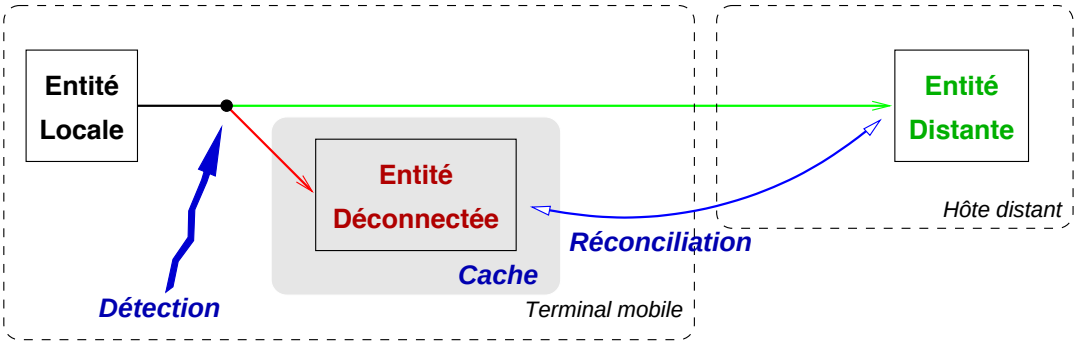
Plan de la présentation

1 Introduction



- Gestion de déconnexion : besoin de continuité de service
  - Déconnexion volontaire : économiser la batterie, minimiser la probabilité de déconnexions inopinées...
  - Déconnexion involontaire : déplacement hors de la zone de couverture du réseau de communication sans fil...

2 Motivations et objectifs de la gestion de cache



- Entité déconnectée pour assurer la continuité de service
- Trois modes de fonctionnement
  - Connecté : opérations effectuées à distance (entité déconnectée non à jour)
  - Partiellement connecté : opérations effectuées localement et à distance
  - Déconnecté : opérations effectuées localement (entité distante non à jour)
- Exécution en mode déconnecté non équivalente à exécution en mode connecté
  - Acceptable si le mode de fonctionnement est explicitement affiché à l'utilisateur

2.1 Rôle du gestionnaire de cache

- Autoriser la participation aux applications réparties : travail collaboratif...
- Tout en exploitant au mieux les ressources locales limitées du terminal mobile
  - Compromis entre :
    - Batterie : traitement en local pour minimiser les communications, voire traitement à distance pour éviter les calculs intensifs
    - Bande passante : traitement en local pour pallier les aléas
    - Mémoire : accès à distance aux bases de données importantes
    - Processeur : traitement à distance des calculs intensifs

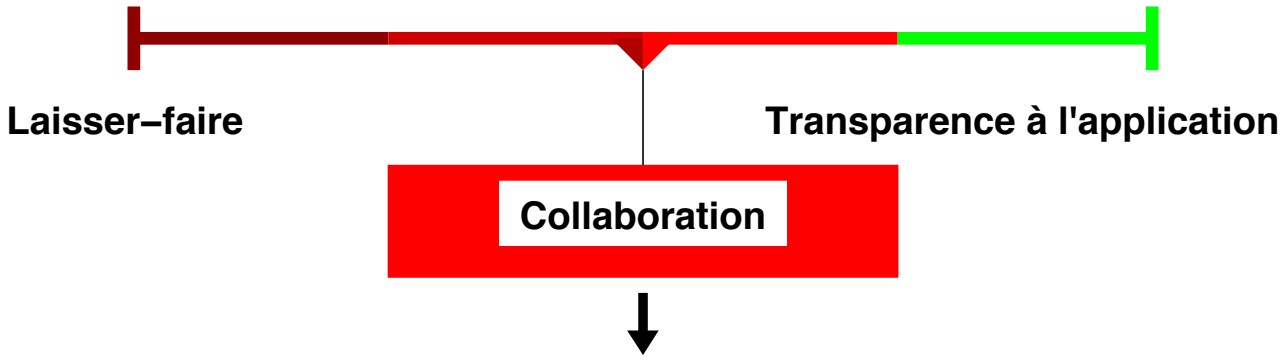
2.1 Rôle du gestionnaire de cache

- Autoriser la participation aux applications réparties : travail collaboratif...
- Tout en exploitant au mieux les ressources locales limitées du terminal mobile
  - Compromis entre :
    - Batterie : traitement en local pour minimiser les communications, voire traitement à distance pour éviter les calculs intensifs
    - Bande passante : traitement en local pour pallier les aléas
    - Mémoire : accès à distance aux bases de données importantes
    - Processeur : traitement à distance des calculs intensifs
- Stratégie de déploiement :
  - À la demande, quelle entité déployer ? Quand déployer ? Où ? Pour quelle durée ?
- Stratégie de remplacement :
  - Au besoin, quelle entité supprimer ? Quand supprimer ? Où ? Pour quelle durée ?
- Stratégie de maintien de la consistance :
  - Cf. la seconde partie de la présentation par Sophie Chabridon

3 Rapide état de l'art

3.1 Stratégie d'adaptation

- [Satyanarayanan, 1996, Capra et al., 2002]



- [Jing et al., 1999] : stratégies « laisser-faire » et « transparence » non adéquates

3.2 Type d'adaptation

- Trois types d'adaptation [Bruneton, 2001]
  - Statique : adaptation durant le développement
  - Dynamique : adaptation durant l'exécution à l'initiative et contrôlée par un acteur extérieur à l'application
    - Acteurs : intergiciel et/ou utilisateur final
      $\Longrightarrow$ Capacité de sensibilité au contexte
      $\Longrightarrow$ Intergiciel avec gestion/détection de contexte
  - Auto-adaptation : adaptation automatique durant l'exécution contrôlée par l'intergiciel

3.3 Moyen d'adaptation

- Réflexivité : auto-introspection et auto-modification
  - Comportementale : réification de l'exécution et des interactions entre entités
    - Traitement pré/post méthode
    - Traitement pré/post requête
  - Structurelle : réification de la structure du système : entité de l'intergiciel ou de l'application
- Dans le domaine de la mobilité :
  [Al-bar and Wakeman, 2001, Capra et al., 2002, Blair et al., 2004]
- Corollaire et objectif de la réflexivité :
  - Meilleure séparation entre les aspects fonctionnels et extrafonctionnels

3.4 Type de cache

- Cache local, donc non partagé avec d'autres terminaux mobiles
  - Un cache unique pour toutes les applications
  - Un cache interapplication (pour toutes les applications) et un cache applicatif par application
- Cache réparti géré de manière centralisée ou répartie
  [Dwyer and Bharghavan, 1997, Chockler et al., 2000, Atzman et al., 2002, Boulkenafed and Issarny, 2003]
  - Maintien de la cohérence des méta-données décrivant le cache réparti en plus du maintien de la consistance des entités du cache
  - Groupes de terminaux mobiles

3.5 Type d'entité déconnectée

3.5.1 Système orienté fichiers

- Coda [Kistler and Satyanarayanan, 1991, Mummert et al., 1995, Satyanarayanan, M., 2002] : Premiers travaux sur les opérations déconnectés
  - Fichiers regroupés en volumes
  - Décomposition en trois phases : accumulation, émulation, réintégration
  - Pour le déploiement, base des fichiers à charger construite par l'utilisateur
  - Politique LRU pour la stratégie de remplacement et à la demande de l'utilisateur
  - Partiellement connecté : validation de cache rapide (volume puis fichiers), réintégration progressive (latence et adaptation de la granularité)
- Ficus/Seer [Kuenning and Popek, 1997] :
  - En utilisant Coda, prédiction d'accès aux fichiers
  - Observation des actions des groupes de processus de l'utilisateur
    - Distance sémantique : cycle de vie plutôt que temporelle ou d'accès
    - Regroupement par projets dynamiques
    - \+ Concept de criticité : de $0 =$ blocage, à $4 =$ non demandé actuellement
    - Durée accumulation : 2 min, surcoût de récupération des stats négligeable
    - Sensibilité aux mauvais choix de l'utilisateur

- Amigos [Andersen et al., 1994] :
  - Réplication pessimiste
  - Extension de NFS
  - Profil utilisateur pour le déploiement : liste de fichiers avec priorité
  - Remplacement périodique avec statistiques d'utilisation (fréquence)
- Université du Michigan, USA [Huston and Honeyman, 1995]
  - Opération partiellement connectée :
    - Modification du gestionnaire de cache de AFS
    - Utilisation du réseau même si faible bande passante
    - Cohérence *Fetch only* : modifications vues localement avant réintégration
    - Classification du trafic réseau et gestion de priorités :
      - ★ Interactions de l'utilisateur, transferts de journaux d'opérations, autres

- Prayer File System [Dwyer and Bharghavan, 1997]
  - Architecture trois tiers : client sur le terminal mobile, client fixe référent et serveur
  - Structuration des fichiers : tout le fichier, certains enregistrements
  - Modification de l'opération de lecture
    - Consistante : erreur ou blocage ou local, locale : données locales
  - Modification de l'opération d'écriture
    - *Through* : erreur ou blocage ou local, *back* : journalisation, local
- Université de Saskatchewan, Canada [Froese and Bunt, 1999, Mei and Bunt, 2003]
  - Simulations en mode partiellement connecté
  - Trois traces du monde académique et changement aléatoire de la bande passante
  - Deux paramètres étudiés :
    - Chargement du fichier en entier ou par blocs
      - Taille « bloc » fonction de la capacité du réseau : gain $\approx 10\%$
      - Accroissement des performances lors des lectures
    - Stratégies de réintégration dynamiques : diminuer la taille du journal et le risque de conflits
      - Agressive : cache 10Mo, réseau 14Kb/s, surcoût 2% des lectures : facteur 5
  - Politique LRU et accumulation pendant 24 heures pour une semaine d'autonomie

3.5.2 Système orienté bases de données

- Bayou [Terry et al., 1995, Petersen et al., 1997]
  - Terminal mobile à la fois client et serveur
  - Propagation des mises à jour avec un protocole anti-entropique
  - Déploiement toute la base de données au lancement de l'application
- Deno[Çetintemel et al., 2003]
  - Chaque copie possède un poids, propagation des votes et validation à la majorité
  - Cohérences faible (mises à jour) ou forte (requêtes et mises à jour)
  - Vote spéculatif : quelque soit l'état des transactions précédentes
  - Possibilité de donner sa délégation de vote à un mandataire
- Rechargement de données [Cherniack et al., 2001]
  - Profil utilisateur en terme de domaines des données
    - Différentes résolution, compression et fidélité
    - Dépendances entre données
  - Préférences en terme de priorité (comparabilité)
  - Agrégation de profils
- Comparaison de protocoles de validation [Bobineau et al., 2004]

3.5.3 Système orienté objets

- Rover [Joseph et al., 1995, Joseph et al., 1997]
  - Mandataires sur le terminal mobile : objets dynamiques relogeables
    - Primitives `import`, `load`, `invoke` et `export`
    - Options d'importation : *uncacheable* (après `export`), inaltérable, à vérifier avant utilisation, vérifier si service disponible, expiration après date, notifier modification par le serveur
    - Informations de connectivité présentées à l'utilisateur
    - Stratégie « collaboration » : conception mode dégradé avec ce langage
  - File d'attente des appels de procédures
    - Ordonnancement : priorité des messages donnée par l'application
- Cascade [Chockler et al., 2000, Atzman et al., 2002]
  - Cache réparti, notion de domaine, réplication structurée en arbre
  - Contrôle par le client CORBA, granularité objet ou requête
  - Options : propagation des écritures, lire ses écritures, ordre FIFO des lectures, ordre causal des écritures, ordre total
  - Étude de politiques de remplacement : LRU, LFU, LFU-DA, SIZE, INVERSE (SIZE), GDSF + 2 nouvelles : H-BASED, LFU-H-BASED

3.5.4 Système orienté paquetages/composants

- Achilles [Kortuem et al., 1997]
  - Déploiement de logiciels à la demande
    - Manuel (choix de l'utilisateur) ou automatique (tous ceux utilisés)
    - Granularité paquetage et graphe de dépendances de ressources entre logiciels
    - Qualification des ressources nécessaires (mémoire, bande passante...)
  - Remplacement manuel ou automatique : coût chargement, nombre de dépendances
  - Affichage des informations de connectivité à l'utilisateur final
- CÉSURE et CADeComp [Bernard et al., 2000, Ayed et al., 2004]
  - Déploiement adaptatif d'applications à base de composants
- Configuration dynamique de téléphones mobiles [Roman and Islam, 2004]
  - Expression des contraintes de déploiement

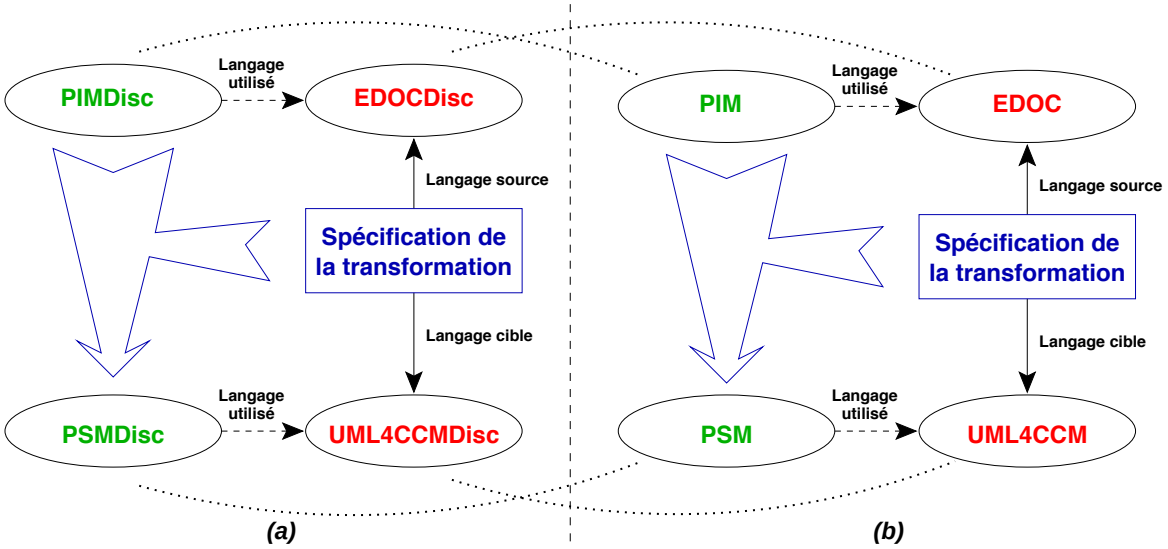
3.6 Positionnement de nos travaux

- Cache de composants pour la gestion de déconnexion
  - Dans cette présentation, étude du déploiement dans le cache et du remplacement des entités du cache
- Stratégie d'adaptation « collaboration » :
  - Méthodologie de développement (MADA) pour l'adaptation statique
    - Processus de développement d'applications à base de composants
  - Canevas logiciel orienté composants (DOMINT) permettant la réflexivité comportementale et structurelle pour l'adaptation dynamique et l'auto-adaptation
- Cache local partagé par toutes les applications du terminal mobile
- Orientation service avec services déconnectés et composants les réalisant
  - Granularité différente et dépendante de l'application
  - Connaissance de la sémantique des opérations

4 Modélisation de la gestion du cache avec MADA

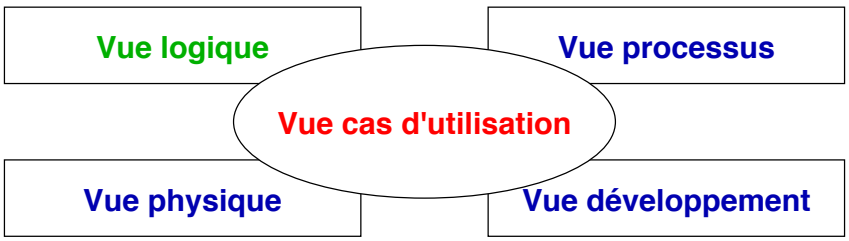
4.1 Approche OMG MDA

- [OMG, 2003]
  - Orientée modèles, centrée sur l'architecture, à base de composants



4.2 Modèle "4 + 1" vues de l'architecture logicielle

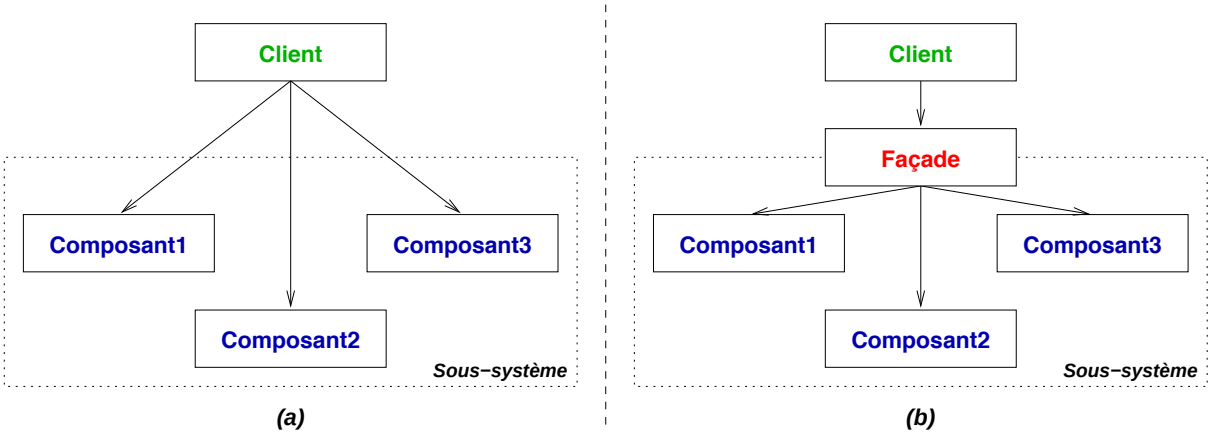
- [Kruchten, 1995]
  - Vue cas d'utilisation : définir les fonctionnalités[a] utilisables en mode déconnecté
  - Vue logique : décrire les fonctionnalités dans des diagrammes de classes et d'interactions
  - Vue processus : exprimer les changements de mode de fonctionnement dans des diagrammes d'activité
  - Vue développement : décrire l'organisation logicielle de l'application dans des diagrammes d'implantation
  - Vue physique : exprimer le déploiement de l'application



[a] Ou service

4.3 Concept de service et patron de conception Façade

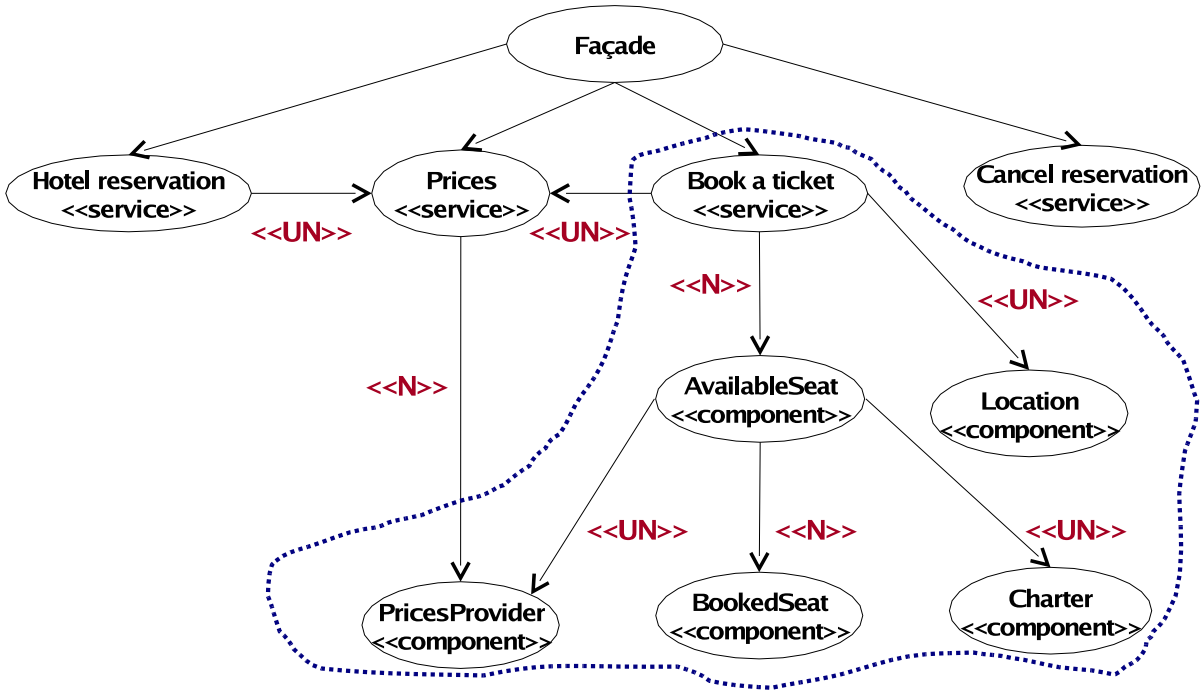
- Service : ensemble logique de composants
  - Granularité de structuration de l'application
  - Identifié dans les diagrammes de cas d'utilisation
- Patron de conception « Façade »
  - Réduire le nombre de composants exposés à l'interface de l'utilisateur final
  - Faciliter la gestion des services et des composants : remplacement...



4.4 Profil de l'application

- Déconnectabilité :
  - Indique si une entité déconnectée de l'entité de l'application de l'hôte distant est autorisée
    - Du point de vue de la sémantique de l'application : par ex., cohérence des données
    - Du point de vue de l'architecture de l'application : par ex., sécurité
- Nécessité :
  - Indique si l'entité déconnectable est obligatoire pour le bon fonctionnement de l'application en mode déconnecté
    - Choix de l'architecte
    - Surcharge possible de ce choix par l'utilisateur de « non nécessaire développeur » vers « nécessaire utilisateur »
- Priorité :
  - Indique l'importance relative de l'entité déconnectée pour le déploiement et le remplacement

4.5 Graphe de dépendances



4.6 Propagation dans le graphe de dépendances

- Règles de gestion du graphe formalisées :
  - Nécessité d'un service $\stackrel{\text{def}}{=}$ nécessité entre la façade et le service, puis propagation depuis un service
  - Nécessité d'un composant $\stackrel{\text{def}}{=}$ nécessité d'au moins un service englobant, puis propagation depuis un composant
  - Évolution dynamique du graphe à partir de changements par l'utilisateur
    - Changement de nécessité d'une fonctionnalité $\stackrel{\text{def}}{=}$ changement de nécessité du lien entre la Façade et le service réalisant la fonctionnalité, puis propagation du changement
- Interface graphique de gestion des méta-données facile à concevoir

5 Plateforme DOMINT

5.1 Stratégie de déploiement

- Au démarrage de l'application sur le terminal mobile
  - Tous les services nécessaires-développeurs
    - Les composants nécessaires-développeurs de ces services
- À la demande de l'utilisateur
  - Lors d'un changement de nécessité par l'utilisateur
- Lors de l'invocation
  - Pour les services nécessaires-utilisateurs et non nécessaires
    - Les composants nécessaires-développeurs de ces services
  - Pour les composants nécessaires-utilisateurs et non nécessaires

5.2 Stratégie de remplacement

- À la demande de l'utilisateur
  - Choix des services à éjecter du cache
- Périodiquement : exécution de l'algorithme suivant

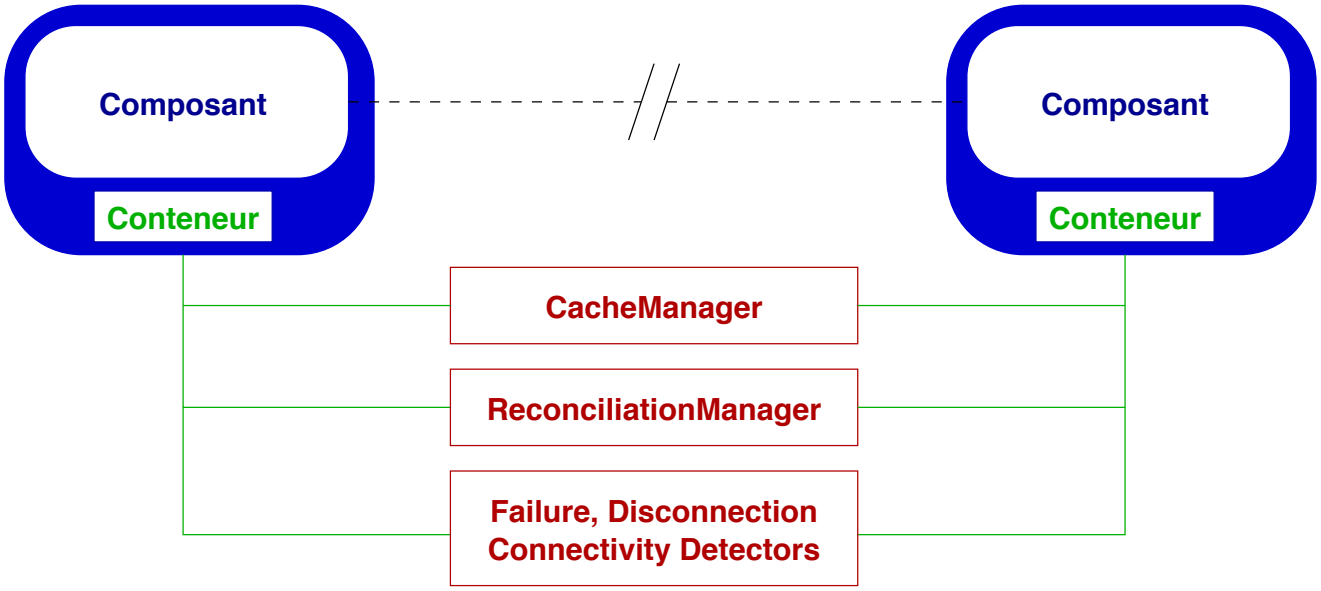
```
1  boolean necessary ← getServiceNecessity(sr)        {true if the service is necessary}
2  string necessityKind ← getNecessityKind(sr)        {"User" or "developer"}
3  DisconnectedComponent dc
4  if necessary and necessityKind="developer"
5        return false                                  {Exit without removing the service}
6  ComponentSet components ← getComponentSet(sr)
7  for all dc ∈ components
8        if (dc.updated and ¬dc.shared) then  removeComponent(dc)
9  removeService(sr)
10 return true                                        {Exit with success after suppression of the service}
```

- Nouvelle politique : *Least Frequently Used with Periodicity and Priority* (LFUPP)
  - Fréquence (LFU) puis priorité et périodiquement réinitialisation de la fréquence

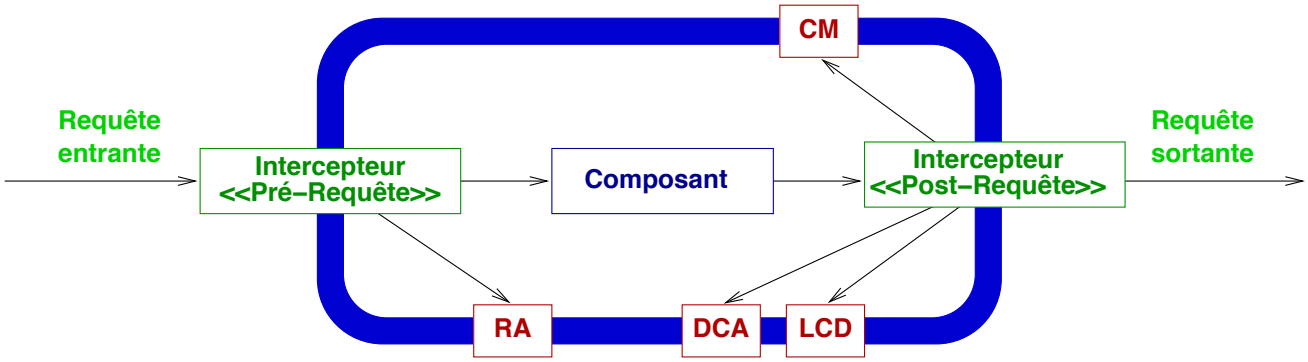
5.3 Architecture de la plateforme

5.3.1 Architecture globale

- Mise en œuvre du patron de conception composant/conteneur [Volter, 2001]
- Modèle de composants CCM [OMG, 2002]
- Canevas logiciel OpenCCM [ObjectWeb, 2004b]



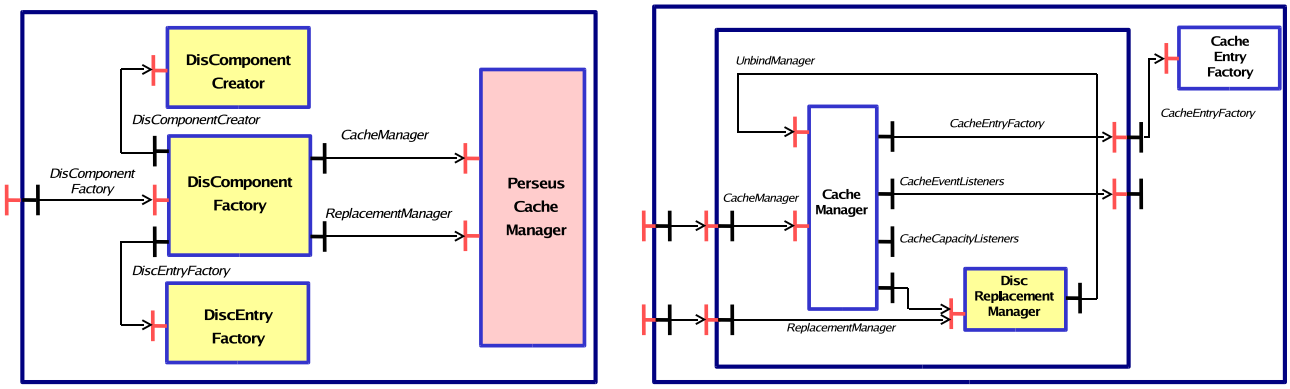
5.3.2 Architecture du conteneur



- DCA : *Disconnected Component Access*
- LCD : *Local Connectivity Detector*
- RA : *Reconciliation Access*
- CM : *Connector Manager*

5.3.3 Architecture du gestionnaire de cache

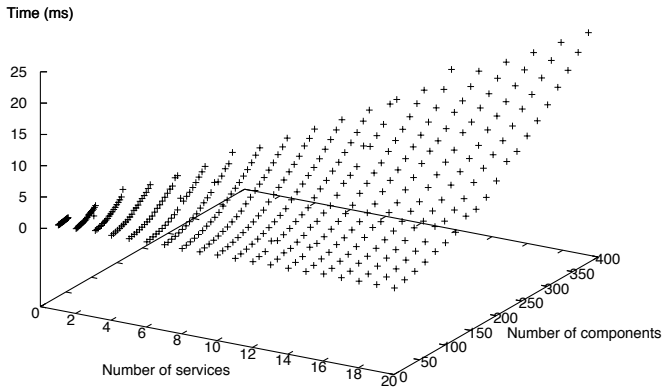
- Modèle de composants Fractal [ObjectWeb, 2004a]
- Utilisation du canevas logiciel Perseus [ObjectWeb, 2004c]



(a) *(b)*

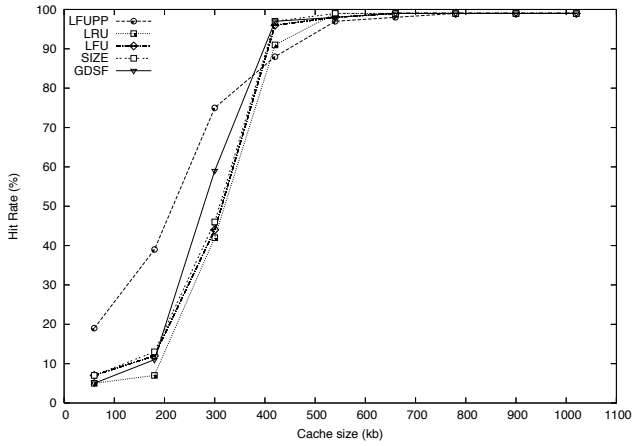
5.4 Mesures de performance

5.4.1 Manipulation du graphe des dépendances

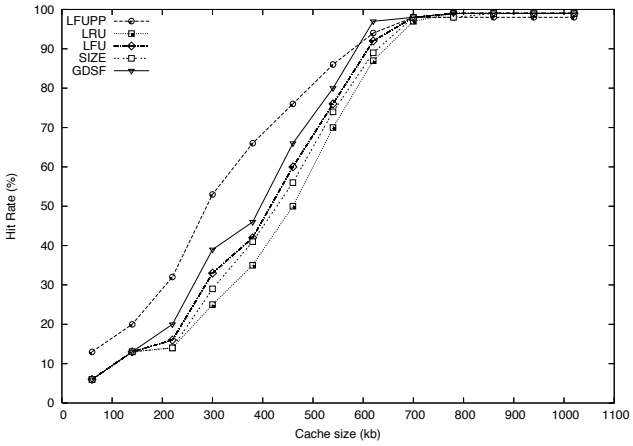


- Durée d'extraction négligeable même en situation extrême :
  - $1.32ms$ pour 1 service et 20 composants
  - $20.2ms$ pour 20 services et 381 composants
- Très peu d'influence du nombre de services

5.4.2 Comparaison des politiques de remplacement



(a) Granularité service



(b) Granularité composant

- Pour les caches de petite taille :
  - Granularité service plus efficiente
  - Politique LFUPP meilleure
- Performances similaires pour les caches plus gros

6 Conclusion et perspectives

- Méthodologie de développement MADA et canevas logiciel DOMINT :
  - Approche MDA, modèle « 4 + 1 » vues de l'architecture, orientation services
  - Concepts de services/composants déconnectés/déconnectables/nécessaires
  - Profil dynamique : graphe de dépendances entre services/composants
  - Conteneur ouvert intégrant les services pour la gestion de déconnexion : par ex., gestion du cache de services/composants déconnectés
  - Déploiement : au démarrage, à la demande de l'utilisateur, lors des invocations
  - Remplacement : à la demande de l'utilisateur, périodiquement
    - Nouvelle politique : LFUPP
  - Canevas logiciels ObjectWeb utilisés : OpenCCM, Fractal, Perseus
- Dernières publications : [Kouici et al., 2004a, Kouici et al., 2004b, Kouici et al., 2004c]
- Projets : ITEA Vivan, RNTL franco-finlandais AMPROS, ITEA Osmose
- Travaux en cours et perspectives :
  - Démonstrateur dans le cadre du projet AMPROS : gestion de crise
  - Détection de partition et gestion de groupes en environnement mobile
  - Partage d'un cache réparti géré de manière répartie par un groupe de terminaux mobiles

Références

[Al-bar and Wakeman, 2001] Al-bar, A. and Wakeman, I. (2001). A survey of adaptative applications in mobile computing. In *Proc. ICDCS Workshop on Smart Appliances and Wearable Computing*, pages 246–251, Mesa, Arizona, USA.

[Andersen et al., 1994] Andersen, B., Jul, E., Moura, F., and Guedes, V. (1994). File System for Semiconnected Operation in AMIGOS. In *Proc. 2nd USENIX Symposium on Mobile and Location-Independent Computing*.

[Atzman et al., 2002] Atzman, H., Friedman, R., and Vitenberg, R. (2002). Replacement Policies for a Distributed Object Caching Service. In Meersman, R. and Tari, Z., editors, *Proc. 4th International Symposium on Distributed Objects and Applications*, volume 2519 of *Lecture Notes in Computer Science*, pages 661–674, University of California at Irvine, USA. Springer-Verlag.

[Ayed et al., 2004] Ayed, D., Taconet, C., and Bernard, G. (2004). Architecture à base de composants pour le déploiement adaptatif des applications multicomposants. In *Actes des Journées Composants*, Lille (France).

[Bernard et al., 2000] Bernard, G., Marvie, R., Putrycz, É., and Taconet, C. (2000). Configuration et exécution de services pour les usagers mobiles des réseaux Étendus. Spécifications de l'infrastructure système, rapport technique du projet RNRT CÉSURE.

[Blair et al., 2004] Blair, G., Coulson, G., and Grace, P. (2004). Research Directions in Reflective Middleware : the Lancaster Experience. In *Proc. 3rd Workshop on Adaptive and Reflective Middleware*, pages 262–267, Toronto, Canada. Middleware 2004 Companion.

[Bobineau et al., 2004] Bobineau, C., Labbé, C., Roncancio, C., and Serrano-Alvarado, P. (2004). Protocoles de validation pour transactions en environnements mobiles : première étude. In *Actes de la 1ère Conférénce ACM Francophone Mobilité et Ubiquité*, pages 86–89, Nice, France.

[Boulkenafed and Issarny, 2003] Boulkenafed, M. and Issarny, V. (2003). A Middleware Service for Mobile Ad Hoc Data Sharing, Enhancing Data Availability. In *Proc. IFIP/ACM/USENIX International Middleware Conference*, volume 2672 of *Lecture Notes in Computer Science*, pages 493–511, Rio de Janeiro, Brazil. Springer-Verlag.

[Bruneton, 2001] Bruneton, E. (2001). *Un support d'exécution pour l'adaptation des aspects non-fonctionnels des applications réparties*. PhD thesis, INPG, Grenoble, France. In French.

[Capra et al., 2002] Capra, L., Blair, G., Mascolo, C., Emmerich, W., and Grace, P. (2002). Exploiting Reflection in Mobile Middleware. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(4) :34–44.

[Çetintemel et al., 2003] Çetintemel, U., Keleher, P., Bhattacharjee, B., and Franklin, M. (2003). Deno : A Decentralized, Peer-to-Peer Object-Replication System for Weekly Connected Environments. *IEEE Transactions on Computers*, 52(7) :943–959.

[Cherniack et al., 2001] Cherniack, M., Franklin, M., and Zdonik, S. (2001). Expressing User Profiles for Data Recharging. *IEEE Personal Communications*, 8(4) :32–38.

[Chockler et al., 2000] Chockler, G., Dolev, D., Friedman, R., and Vitenberg, R. (2000). Implementing Caching Service for Distributed CORBA Objects. In Sventek, J. and Coulson, G., editors, *Proc. IFIP/ACM/USENIX International Middleware Conference*, volume 1795 of *Lecture Notes in Computer Science*, pages 1–23. Springer-Verlag.

[Dwyer and Bharghavan, 1997] Dwyer, D. and Bharghavan, V. (1997). A mobility-aware file system for partially connected operation. *ACM Operating Systems Review*, 31(1) :24–30.

[Froese and Bunt, 1999] Froese, K. and Bunt, R. (1999). Cache Management for Mobile File Service. *The Computer Journal*, 42(6) :442–454.

[Huston and Honeyman, 1995] Huston, L. and Honeyman, P. (1995). Partially Connected Operation. In *Proc. 2nd USENIX Symposium on Mobile and Location-Independent Computing*, Ann Arbor, Michigan, USA.

[Jing et al., 1999] Jing, J., Helal, A., and Elmagarmid, A. (1999). Client-Server Computing in Mobile Environments. *ACM Computing Surveys*, 31(2) :117–157.

[Joseph et al., 1997] Joseph, A., Tauber, J., and Kaashoek, M. F. (1997). Mobile Computing with the Rover Toolkit. *IEEE Transactions on Computers*, 46(3) :337–352.

[Joseph et al., 1995] Joseph, A. D., deLespinasse, A. F., Tauber, J. A., Gifford, D. K., and Kaashoek, M. F. (1995). Rover : A Toolkit for Mobile Information Access. In *Proc. 15th ACM Symposium on Operating Systems Principles*, pages 156–171.

[Kistler and Satyanarayanan, 1991] Kistler, J. and Satyanarayanan, M. (1991). Disconnected Operation in the Coda File System. In *Proc. 13th ACM Symposium on Operating Systems Principles*, Pacific Grove, USA.

[Kortuem et al., 1997] Kortuem, G., Fickas, S., and Segall, Z. (1997). On-Demand Delivery of Software in Mobile Environments. In *Proc. IPPS Workshop on Nomadic Computing*, Geneva, Switzerland.

[Kouici et al., 2004a] Kouici, N., Conan, D., and Bernard, G. (2004a). Caching Components for Disconnection Management in Mobile Envrionments. In Z. Tari *et al*, editor, *Proc. 6th International Symposium on Distributed Objects and Applications*, volume 3291 of *Lecture Notes in Computer Science*, pages 1322–1339, Agia Napa, Cyprus. Springer-Verlag.

[Kouici et al., 2004b] Kouici, N., Conan, D., and Bernard, G. (2004b). Intégration d'un service de gestion des déconnexions dans les conteneurs des composants. In *Actes des Journées Composants*, Lille, France.

[Kouici et al., 2004c] Kouici, N., Conan, D., and Bernard, G. (2004c). MADA : une approche pour le développement d'applications mobiles. In *Actes de la 1ère Conférénce ACM Francophone Mobilité et Ubiquité*, pages 78–85, Nice, France.

[Kruchten, 1995] Kruchten, P. (1995). The 4+1 View Model of Architecture. *IEEE Software*, 12(6) :42–50.

[Kuenning and Popek, 1997] Kuenning, G. and Popek, G. (1997). Automated Hoarding for Mobile Computers. In *Proc. 16th ACM Symposium on Operating Systems Principles*, pages 1–22, Saint Malo, France.

[Mei and Bunt, 2003] Mei, J. and Bunt, R. (2003). Adaptive File Cache Management for Mobile Computing. In *Proc. 4th International Conference on Mobile Data Management*, volume 2574 of *Lecture Notes in Computer Science*, pages 369–373. Springer-Verlag.

[Mummert et al., 1995] Mummert, L. B., Ebling, M. R., and Satyanarayanan, M. (1995). Exploiting Weak Connectivity for Mobile File Access. In *Proc. 15th ACM Symposium on Operating Systems Principles*, pages 143–155.

[ObjectWeb, 2004a] ObjectWeb (2004a). ObjectWeb Fractal home page. http://fractal.objectweb.org.

[ObjectWeb, 2004b] ObjectWeb (2004b). ObjectWeb OpenCCM home page. http://openccm.objectweb.org.

[ObjectWeb, 2004c] ObjectWeb (2004c). ObjectWeb Perseus home page. http://perseus.objectweb.org.

[OMG, 2002] OMG (2002). CORBA Components. Version 3.0. OMG Document formal/02-06-65, Object Management Group.

[OMG, 2003] OMG (2003). MDA Guide Version 1.0. Omg document, omg/2003-05-01, Object Management Group.

[Petersen et al., 1997] Petersen, K., Spreitzer, M. J., Terry, D. B., Theimer, M. M., and Demers, A. J. (1997). Flexible Update Propagation for Weakly Consistent Replication. In *Proc. 16th ACM Symposium on Operating Systems Principles*, pages 288–301, Saint Malo, France.

[Roman and Islam, 2004] Roman, M. and Islam, N. (2004). Dynamically Programmable and Reconfigurable Middleware Services. In Jacobsen, H.-A., editor, *Proc. IFIP/ACM/USENIX International Middleware Conference*, volume 3231 of *Lecture Notes in Computer Science*, pages 372–396, Toronto, Canada. Springer-Verlag.

[Satyanarayanan, 1996] Satyanarayanan, M. (1996). Mobile Information Access. *IEEE Personal Communications*, 3(1) :26–33.

[Satyanarayanan, M., 2002] Satyanarayanan, M. (2002). The Evolution of Coda. *ACM Transactions on Computer Systems*.

[Terry et al., 1995] Terry, D. B., Theimer, M. M., Petersen, K., and Demers, A. J. (1995). Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *Proc. 15th ACM Symposium on Operating Systems Principles*, pages 172–183.

[Volter, 2001] Volter, M. (2001). Server-side Components - A Pattern Language. In *Proc. 6th European Conference On Pattern Languages of Programs*, Irsee, Germany.