Projet RICM4
Application Distribuée

J. Briat, R. Revire, O. Richard

Résumé

Le projet porte sur la mise en place d'un service distribué : une gestion d'un ensemble de bases documentaires. Il sera fait par des groupes de 4 personnes maximum sur une durée de 4 semaines

1 La Bibliothéque : un service distribué

Une Bibliothèque est une application distribuée de gestion de documents "électroniques". Elle maintient une base de données textuelles c'est à dire un ensemble de documents définis comme une paire <*Résumé, Contenu*>. La bibliothèque est organisée en un réseau d'*agences* situées sur différents sites. Une agence n'a qu'une connaissance partielle de la bibliothèque ; elle ne connaît que les documents qu'elle gère et des agences *voisines*.

Les clients contactent une agence pour y déposer des documents, en consulter ou en retirer. Pour ce faire un client ne contacte qu'une agence. Celle-ci se charge des recherches nécessaires dans son fond propre et coordonne les recherches dans les fonds des autres agences si nécessaire.

Chaque agence gère une partie de la base des documents. Une agence ne possède pas tous les documents et un document peut exister en un nombre quelconque d'exemplaires dans une agence ou plusieurs d'entre elles.[1] La classification des documents est locale à une agence et est basée sur une relation d'ordre quelconque (par ex. l'ordre alphabétique). Le critère de répartition des documents entre agences correspond à une politique d'équilibrage de charge (par ex. un nombre de documents équivalent, un volume de caractères équivalent). Comme les retraits et dépots de documents déséquilibrent les fonds documentaires des agences, il faut les reéquilibrer. Périodiquement, les agences évaluent leurs charges et les rééquilibrent si nécessaire en déplaçant des documents. Le choix des documents à déplacer est laissé aux agences.

Les agences doivent donc collaborer en permanence pour satisfaire les requêtes des clients qui demandent à consulter des documents, à en retirer ou à en ajouter selon un protocole transactionnel décrit par la suite.

1.1 Localisation d'un service

Le mécanisme de localisation d'un service distribué[2] est une simple extension du mécanisme de résolution de nom. Un processus client peut localiser un serveur du service en interrogeant un service **Annuaire bien connu** de tous les clients. Ce service Annuaire connaît tous les services disponibles dans le système distribué et tous les serveurs les réalisant. L'annuaire indique le serveur(adresse IP et port) se trouvant sur le site le plus proche du client. Un client localisera un service via la classe et méthode suivante[3] :

```
class Annuaire{
AdresseServeur  localise(String nomDeService){.....}
}
```

1.2 Le Service Bibliothèque

Une Agence peut être amenée à servir plusieurs clients directs ou indirects. En effet, un client ne communique qu'avec l'agence indiquée par l'Annuaire au début de sa transaction. L'organisation du réseau d'agences lui est inconnue (et doit le rester).

Une fois une agence localisée, le client lui soumet les diverses requêtes d'une transaction. Un client peut engager plusieurs transactions à la fois sur une Bibliothèque. Trois types de transaction permettent de consulter, retirer et ajouter des documents. Une transaction de type consultation ou retrait commence toujours par une phase de sélection. Dans la phase de sélection, des documents sont sélectionnés à partir d'un filtre de recherche. Celui-ci est une liste de chaînes de caractères. Un document est sélectionné si l'une des chaînes apparaît dans le résumé. L'étape suivante est une étape de consultation ou une étape de retrait de documents

[1] Chaque base partielle sera **simulée** en mémoire par un simple tableau de chaînes de caractères (une pour le résumé, l'autre pour le contenu).
[2] C'est à dire comportant plusieurs serveurs
[3] A définir et réaliser

sélectionnés. Une requête de sélection porte sur **toute** la bibliothèque. La requête doit donc être acheminée vers les autres agences depuis l'agence initiale a l'insu du client. Les documents sélectionnés peuvent alors être consultés ou bien retirés définitivement.

Des documents peuvent être ajoutés à la bibliothèque par une transaction de dépôt (qui ne comporte pas d'étape de sélection).

1.3 L'interface Client

Les protocoles de communication et les détails de fonctionnement du service bibliothèque doivent être masqués au client. Un processus client invoque ce service par des procédures "subrogées" conformes aux modèles ci-après et qui constituent l'Interface Applicative (API) du service Bibliothèque. La réalisation de cette API doit être opaque pour les clients.

```
class Document{
   String résumé;
   String contenu;
}
class TransactionBibliothèque {
   // type de transactions
   protected static int RETRAIT ;
   protected static int CONSULTATION ;
   protected static int DEPOT ;

  TransactionBibliothèque(AdresseServeur ,int );
  int réserve( String[]);   // réserver des documents ;
                             // rend le nombre réservé
  Document lire(int);        // lit contenu et résumé
                             // du nième document
  void libére (int);         // libère le nième document
  void valide () ;           // valide ou
  void invalide() ;          // invalide la transaction
  void dépot(Document[]);    // dépot de documents+résumés
}
```

Initialisation Un objet transaction est créé en donnant son type et le serveur agence à contacter. Il encapsule les informations nécessaires pour authentifier la légalité des requêtes ultérieures comme pour communiquer avec l'agence sélectionnée.

Sélection Un client peut émettre plusieurs transactions vers le même service. Mais une transaction ne permet qu'une **unique** étape de sélection. Le filtre de documents est un tableau de chaînes de caractères. Un document est sélectionné si une des chaînes est une sous-chaîne du résumé du document. Tous les documents sélectionnables sont sélectionnés. Ce nombre est rendu en résultat de la sélection.

Consultation Que la transaction soit de consultation ou de retrait, le client consulte un document sélectionné par son numéro de sélection c'est à dire un nombre allant de 0 au nombre de document sélectionnés -1.

Libération anticipée Comme la sélection par filtre peut n'être pas assez discriminante, le client peut libérer un document décidé inutile par son numéro de sélection.

Retrait Un document sélectionné par une transaction de retrait peut être retiré du fond d'une agence à l'aide de son numéro de sélection.

Ajout Dans le cas d'une transaction d'ajout, il n'y a pas d'étape de sélection. Le client ajoute des documents dans la base de l'agence avec laquelle il est en contact. Il fournit un tableau de résumés et contenus.

Validation/Invalidation Enfin, toute transaction doit se terminer par une validation ou invalidation. Sur une validation, tous les documents réservés dans la transaction sont libérés et / ou les opérations de la transaction (ajout / retrait) prennent effet. Lors d'une invalidation, tous les documents réservé dans la transaction sont libérés et les modifications des fonds de la bibliothèque ne sont pas prises en compte.

2 Les Problèmes d'algorithme

2.1 Comportement erroné

Toute erreur (état de transaction, numéro de document incorrect, requête illégale, déconnexion d'un client avant fin de transaction, transaction trop longue, etc..) est interprétée comme une fin de ransaction avec invalidation.

2.2 Synchronisation

Des synchronisations sont nécessaires pour régler les conflits d'accès aux documents de la bibliothèque.

Monopolisation Un client ne peut sélectioner et monopoliser les documents de la bibliothèque. Ils seront automatiquement récupérés par les agences au bout d'un délai FIXE.

Exclusion entre transactions La construction du filtre est une opération locale au client. Dans la phase de sélection, les transactions sont en conflit pour l'acquisition de documents sélectionnables. La politique à appliquer est une variation de la politique lecteur-rédacteur. Un document ne peut être sélectionné pour retrait potentiel que par un client à la fois. Ce n'est pas le cas pour des sélections pour consultation qui peuvent consulter en parallèle un document. Lorsque deux transactions sont en conflit sur un document, une doit attendre la fin de l'autre.

En conséquence, une transaction de consultation et une transaction de retrait ne peuvent avoir lieu en parallèle que si elles portent sur des ensembles disjoints de documents sélectionnables. Les transactions de consultation peuvent toujours avoir lieu en parallèle.

Une transaction d'ajout porte localement sur une agence. On peut réaliser une telle transaction simplement de façon localement exclusive. Les réorganisations des fonds des agences doivent être faites en exclusion mutuelle de toutes transactions.

Interblocage des transactions Le problème principal de ces synchronisations est l'interblocage du fait des réservations des documents par des agences différentes ou des ordres différents.. On règlera ce problème par une stratégie d'allocation **ordonnée** libre d'interblocage. Une première stratégie consiste à garantir que les documents sont acquis dans un ordre identique pour toutes les transactions. La seconde est d'utiliser une stratégie de type WAIT OR DIE basée sur un ordre des transactions Ces stratégies doivent être sans famine. Les transactions qui ne peuvent acquérir un document sélectionnable doivent être suspendues. <u>Toute solution d'attente active est interdite</u>.

Retrait Les retraits de documents ne sont effectifs qu'à la fin de la transaction de retrait (validation).

2.3 Communications

Les Agences interagissent en permanence entre elles et avec les clients.

Circulation des requêtes clients Les contacts agence - client se font toujours via l'agence initialement contactée. Une agence retransmet donc vers sa(ses) voisine(s) les requêtes qu'elle ne peut satisfaire seule. Les requêtes de sélection doivent atteindre toutes les agences concernées en circulant dans un réseau logique de connexion (réseau d'agences). On prendra garde au problème d'interblocage des communications. Une agence sera émetteur et récepteur vis à vis des autres agences et de ses clients.

Abandon de transactions Une agence contrôle la légalité des transactions des clients qui lui sont directement connectés.Elle est responsable de la détection des conditions d'invalidation forcée d'une transaction. Elle propage les fins de transactions normales ou brutales (erreur) aux autres agences.

2.4 Réorganisation des bases locales

La réorganisation des bases locales peut se faire à tout moment à l'initiative d'une agence détectant une surcharge ou souscharge induite par des opérations de dépôt ou de retrait. Cette réorganisation doit se faire à l'insu des clients. A la détection d'un déséquilibre, les agences doivent se mettre d'accord sur une nouvelle répartition. Cette nouvelle répartition se traduit par une redistribution des documents suivie du redémarrage du fonctionnement normal. Cette réorganisation peut se faire de façon exclusive de toutes transactions ou seulement des transactions d'ajout / retrait.

3 Les problème d'architecture

Ils concernent essentiellement l'installation du réseau de communication entre agences, entre clients et agences, entre clients et Annuaire et entre agences et Annuaire.

3.1 Annuaire

Dans un système distribué, celui-ci a pour rôle de donner à tout client l'adresse IP et le port "bien connu" d'un serveur apte à réaliser un service. Dans notre cas, ce serveur fournira l'adresse IP et le port du serveur "le plus proche" du client. Il lui faut donc connaître ces serveurs.

Dans la solution demandée, les serveurs doivent s'enregistrer comme serveur d'un service donné (par ex. le service BIBLIOTHEQUE) en fournissant leur adresse IP et un port de connexion. Un service ne sera complètement accessible aux clients qu'à partir du moment où tous les serveurs prévus se sont enregistrés. Un fichier ˜/SERVICES contiendra les noms des services possibles du système distribué et le nombre de serveurs devant le réaliser au cours du test considéré. L'annuaire doit préexister aux clients et serveurs. Le serveur Annuaire réalisé devra être capable de gérer plusieurs services en même temps.

Enregistrement des serveurs Tout serveur débute par son enregistrement auprès du serveur Annuaire en fournissant le service auquel il participe, son adresse IP et son port de connexion. Cette opération est bloquante jusqu'à ce que l'Annuaire obtiennent la table de placement (adresses IP et Ports "bien connus") du nombre indiqué de serveurs réalisant le service. Il donne alors à chacun des serveurs la localisation de ses voisins au sein du service considéré.

Interrogation d'un client L'annuaire indique le serveur se trouvant sur le site le plus proche du client. On pourra prendra comme métrique la relation d'ordre sur les adresses IP. On prendra garde à la situation où un client se présente de façon prématurée pour utiliser un service non encore déclaré ou valide.

3.2 Réseau d'agences

Les agences sont amenées à coopérer entre elles de façon permanente pour la réalisation des requêtes des clients et pour la répartition du fond de bibliothèque. Les sites sont considérés comme fiables mais non le réseau. Les Agences correspondront via des liaisons TCP.

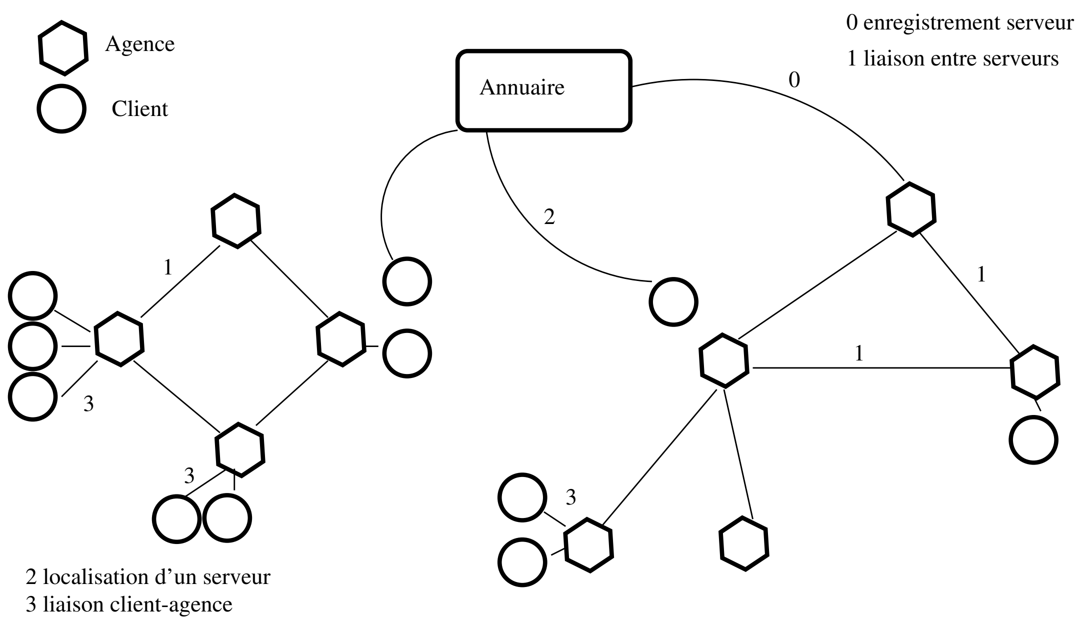

FIG. 1 – Deux exemples de réseau d'agences

Graphe On considère que le nombre potentiel d'agences interdit qu'une agence puisse connaitre toutes les autres agences d'un service[4]. Comme les requêtes auprès de la bibliothèque exigent la coopération de toutes les agences, il sera nécessaire de faire circuler la requête d'un client depuis une agence à toutes les agences (vague) au sein du réseau défini par la relation de *voisinage*

[4]Une raison peut être qu'un Annuaire ne connait qu'une partie des agences ou que le nombre d'agences d'un service varie dans le temps

des agences. Deux approches sont possibles : laisser à ce réseau une structure de graphe quelconque ou lui donner une structure géométrique particulière (anneau mono ou bidirectionnel, grille 2D, etc..).

Liaison Pour des raisons de fiabilité, il est nécessaire d'utiliser des liaisons TCP entre agences. Cependant ces liaisons peuvent être établies de façon statique à l'installation du service[5] ou dynamiquement à la demande lorsqu'une requête doit être transmise à une agence voisine. Les liaisons avec un client sont établies pour la durée d'une transaction.

Le projet porte donc sur le service BIBLIOTHEQUE. Tout test commencera par le lancement de ce serveur et la récupération de son port "bien connu".

4 Organisation logicielle

Langage de programmation La programmation se fera en Java ou en C/C++ + Posix Thread. Ils présentent tous les mécanismes nécessaires (socket , thread). Ce sont des langages que tout ingénieur système+réseau doit maîtriser. Le choix du langage est libre. Toutefois, du fait de la plus grande facilité d'utilisation de Java (ramasse-miette automatique, compilateur plus méfiant, bibliothèque de structures de données plus fournies.,bibliothèque graphique plus souple), un bonus sera donné à l'usage de C/C++ sur l'usage de Java.

Architecture des serveurs : itératif / maître - esclave Les possibilités d'architecture d'un serveur agence sont les suivantes :

1. un serveur itératif complexe
2. un serveur maître / esclaves multi-threads C-Posix ou Java
3. un serveur maître - esclaves multiprocess UNIX

La première solution revient à concevoir un automate non déterministe entrelaçant l'écoute passive des connexion des clients, le traitement de requêtes locales et le traitement des requêtes non locales. Dans un serveur concurrent, un maître distribue les requêtes à des esclaves qui doivent se synchroniser sur l'accès aux données communes. Un serveur maître-esclave est donc facile à faire en java ou C/C++ + Posix Thread (implantation dite légère).

Dans la troisième solution, les esclaves sont des process UNIX (créés par fork(). Leur coopération vis a vis des documents locaux nécessite de placer ceux-ci en mémoire partagée(Shared Memory SystemV ou Berkeley) et de synchroniser (Sémaphores SystemV) leurs accès. Par ailleurs, l'esclave doit relayer les requêtes vers d'autres agences ce qui implique une synchronisation supplémentaire sur les liaisons TCP entre agences ou l'ouverture de liaisons spécifiques. Cette architecture est interdite du fait de sa lourdeur, du nombre de synchronisations, connexions requises et des concepts et outils nouveaux à maîtriser.

La seconde solution permet le partage de mémoire et la synchronisation sans problème technique UNIX. Un maître se contente de créer les esclaves nécessaires au traitement de toute nouvelle transaction. Toutefois les sockets ne peuvent être utilisées sans précaution par les threads si ces sockets sont partagées (problème de l'atomicité d'émission/réception de messages)

La première solution nécessite l'écriture d'un automate communicant non bloquant ce qui est toujours difficile. L'utilisation de threads facilite la réalisation si l'on maîtrise la synchronisation. Dans le premier cas, la mise au point se fait dans un cadre homogène et séquentiel (au sein d'une agence). Dans le second cas, on doit faire face au non déterminisme d'exécution des threads. Globalement, la complexité du problème reste identique ; il y a simplement déplacement de difficultés. A ce jour, l'expérience a montré que l'usage de threads a facilité la réalisation.

Organisation du programme agence Quelle que soit l'architecture choisie, on aura intérêt à organiser le programme en trois parties :

1. une partie responsable de l'organisation des liaisons, de la gestion équitable et non bloquante des communications. Elle est organisée autour des sockets de connexion, des liaisons avec les clients, des liaisons avec les agences et de l'arbitrage des accès aux liaisons partagées.
2. une partie responsable des relations avec les clients direct(vérification des requêtes, respect des délais etc..) et les agences voisines,
3. une partie responsable des accès aux documents. Son point clef est le catalogue des documents, l'état de ceux-ci et des files de transactions suspendues.

[5] immédiatement après enregistrement auprès de l'Annuaire

5 Mise en oeuvre

5.1 Port bien connu

Tous les ports peuvent être allouées dynamiquement même le port bien connu de l'annuaire. Il suffira de lancer celui-ci au préalable et d'obtenir son port. Les clients et agences sont lancés ensuite avec ce port en paramètre.

5.2 Mise en place du réseau des agences

On effectuera d'abord la mise en place du squelette de l'application du point de vue de la communication.

Annuaire : définition du serveur Annuaire , des procédure d'enregistrement des serveurs d'un service et des procédure d'interrogation par un client pour obtenir le serveur "le plus proche" pour un service donnée. Ce serveur Annuaire doit être "général" en ce sens qu'il doit être à même de gérer plusieurs services.

Interconnexion des Agences : réaliser l'interconnexion de tous les agences selon la géométrie choisie si le choix est d'installer au lancement du service les liaisons TCP entre agences .

Connexion d'un client et requête *vide* (ping) : Il s'agit de connecter un client à une agence qui initie une **vague** appropriée (cf prévention de l'interblocage) pour atteindre toutes les agences et acquérir leurs réponses. A la suite de quoi, le client se déconnecte.

5.3 Prévention de l'interblocage

Ce problème a une incidence forte sur les protocoles de communication entre les interlocuteurs (agences, clients) et les structures de données propres à une agence (son état local) :

- liste des documents et état de réservation / blocage. La validation retardée des retraits nécessite de conserver les documents retirés jusqu'à la fin de transaction.
- la liste des clients connectés et l'état de leur transaction (requête en cours, bloqué, etc..).
- la liste des requêtes de clients distants transmises par d'autres agences et état.

Cette définition se fera à partir de l'étude de la circulation(vague) des requêtes issues des clients, entre les agences et la mise en évidence des conditions interdisant la progression d'une requête (document demandé mais déjà réservé, etc..). Le principal problème sera d'éviter les interblocages :

- dues aux communications entre les différents processus via les liaisons TCP. Ce premier type de blocage s'évite en utilisant systématiquement des constructions de communication garantissant que si une communication parmi plusieurs est possible, on pourra la faire. Dans le cadre d'un serveur itératif, ceci se traduit par un usage systématique du `select()` pour déterminer les liaisons TCP non bloquantes. Dans le cas d'un serveur concurrent, un ou plusieurs threads dédiés aux communications sont nécessaires.
- dues aux synchronisation entre les transactions initiées par les clients (consultation, retrait et ajout) et celles initiées par les agences (répartition de charge).

Comme dit précédemment, ceci nécessite la mise en place d'une stratégie d'allocation des documents sans interblocage. Deux stratégies sont possibles. La première est une stratégie *d'allocation ordonnée* réservant les documents selon un ordre unique. Ceci nécessite un algorithme de **parcours** du réseau des agences c'est à dire une vague parcourant les agences selon un ordre donné fixe (ex. 0 puis 1 puis ...) [6]. Un tel choix conduit naturellement à une structure d'*anneau*. La seconde stratégie est une stratégie optimiste du type WAIT OR DIE qui réserve les documents si possible. En cas de rencontre d'un document déjà réservé, la transaction courante se base sur la comparaison des estampilles temporelles des 2 transactions pour décider si elle se bloque en attente de libération du document ou effectue un "retour arrière" c'est à dire libère les documents préalablement réservés lors de la visite d'autres agences[7]. La transaction avortée pourra reprendre après terminaison de la transaction gagnante. Il est alors possible d'utiliser un réseau quelconque(et vague correspondante). L'équité de cette stratégie est assurée par l'utilisation d'une datation des transactions soit par une horloge logique (+ recalage) soit pas une horloge physique de finesse suffisante pour éviter les vols.

On remarquera que toutes les interactions entre agences se ramènent à un problème unique de réservation de ressources. En effet, le rééquilibrage doit réserver tous les documents de la bibliothèque.

L'absence de famine nécessite d'éviter qu'un client ne monopolise les livres réservés. Pour ce faire la durée de réservation est bornée. Il appartient à une agence de contrôler ses clients et de forcer une fin de transaction avec invalidation en cas de dépassement du délai.

[6] et chaque base locale est inspectée selon un même ordre (lexicographique par ex.)

[7] Si la transaction courante est la plus ancienne, elle attend sinon elle abandonne

5.4 Réorganisation des bases locales

La réorganisation des bases locales peut se faire à tout moment à l'initiative d'une agence détectant une surcharge ou une sous charge. Cette réorganisation doit se faire à l'insu des clients. A la détection d'un déséquilibre local (dépassement de seuil haut ou bas), les agence doivent se synchroniser (arrêter les transactions en cours ou attendre leur terminaison) et élire un représentant chargé de calculer une nouvelle partition. A la suite de ce calcul, les documents à déplacer sont déplacés et les transactions sont reprises à partir d'un nouvel état de charge et de nouveaux seuils. Cette réorganisation peut se faire en exclusion de toutes transactions (méthode simple) ou des seules transactions d'ajout et de retrait(méthode complexe) . Ceci peut conduire à un algorithme complexe car autoriser durant l'équilibrage les phases de sélection / consultation nécessite de savoir à tout moment si un document inspecté a déjà été vu ou non par la requête. En effet, les déplacements de document peuvent s'entrelacer avec les sélections.

6 Rendu et Notation

Le document rendu sera un dossier technique comprenant :
- une description précise de l'interface applicative client (types et profil de fonction),
- une description synthétique de l'architecture du système bibliothèque (processus, fonctions, interactions...)accompagnée de la présentation et de la justification des algorithmes distribués réalisés[8] ainsi que des principales décisions techniques,
- une brève description de l'organisation et du découpage des programmes (et fichiers associés),
- le code commenté et makefile associé,
- le point sur l'état d'avancement et l'intégration des différentes parties,
- la description des tests effectués et démonstrations proposées.

Le projet fera l'objet d'une présentation orale (+projection de transparents) et **SURTOUT** d'une démonstration de l'état d'avancement du "produit". La démonstration s'attachera à montrer :
- la bonne installation du réseau d'interconnexion et la circulation des requêtes au sein des agences,
- le bon fonctionnement de l'équilibrage entre agences par des transactions de gestion déséquilibrant la répartition des types de documents,
- la résistance à l'interblocage dans le cas de transactions en conflit (ainsi que la dégradation deperformances avec le taux de conflit),
- le niveau d'intégration.

Une façon de mettre en évidence les comportements des acteurs (annuaire, agences, clients) est de visualiser dans une fenêtre par acteur un résumé graphique de son état et de sa progression. Il est conseillé d'investir dans un petit langage de commande client permettant d'avoir un client interactif (les tests en sont facilités et s'étendent aisément en démonstration).
Une condition nécessaire à l'obtention de la moyenne est de faire une démonstration du fonctionnement d'une partie significative du système.
ATTENTION : Le nombre de problèmes et leurs interactions font échouer A PRIORI une démarche linéaire ramenant le projet à une étape d'analyse-décomposition suivie d'une étape de programmation individuelle de modules suivie d'une étape d'intégration-test. Les groupes doivent procéder par développement incrémental. C'est à dire, après analyse, la réalisation doit s'organiser en étapes successives de *<développement-intégration-test>* permettant de s'assurer chaque fois du fonctionnement correct d'un système de plus en plus complet. Ceci permettra toujours de montrer un état d'avancement du projet. La définition de ces étapes fait partie intégrante de l'analyse du problème et est liée à la définition de l'architecture du système.
NOTE : Les groupes devront résister systématiquement à la tendance à proposer des solutions compliquées à des problèmes simples ou à rajouter des problèmes supplémentaires aux problèmes proposés. Entre autres, l'organisation de la base de documents n'a aucune importance[9]. C'est un simple tableau de chaînes de caractères en mémoire dans lequel on peut sélectionner des éléments sur un critère simple (égalité de chaîne, préfixe, inclusion bref les procédure des bibliothèques standards de langages C/C++ ou Java) .

7 Extension

Le projet est un projet d'application distribuée. Par conséquent, si des extensions sont proposées par un groupe, celles-ci doivent correspondre à ce thème. Aussi, plutôt qu'investir dans des interfaces graphiques sophistiquées mais sans rapport avec le sujet, il vaut mieux se pencher sur les points suivants :
- Ajout / Retrait dynamique d'agences à la Bibliothèque ,
- Problème de fiabilité : cassure des liaisons des clients, des agences.

[8] en particulier, le coût en nombre de communication et étapes.
[9] Ainsi que le fait qu'un même document existe en plusieurs exemplaires.