MASTER EN SCIENCES INFORMATIQUES

Conception d'un système fédéré dynamique d'utilisation de ressources bioinformatiques hétérogènes

Debaisieux, Laurent; De Souza, Fernando

Award date:
2003

Avant-Propos

Ce travail est l'achèvement de quatre années de travail et sacrifices. Ces études furent longues, parfois pénibles mais également source de beaucoup de richesses. J'ai rencontré au cours de ces années une équipe enseignante motivée, dévouée et délivrant un enseignement de qualité. Mes remerciements vont à toutes ces personnes et particulièrement au Prof. Englebert. Il a su nous communiquer sa passion, sa finesse et sa rigueur. Ses conseils et remarques, sa vision toujours claire et ses encouragements nous ont beaucoup aidé.

Ce travail a été réalisé avec la collaboration du Prof. Colet et son équipe, Valérie Ledent et Daniel Van Belle de l'Unité de Bioinformatique de l'Université Libre de Bruxelles. Qu'ils soient remerciés pour leurs éclaircissements, leur disponibilité et leur gentillesse. Leur aide fut précieuse pour mettre les pièces du puzzle en place.

A mes « compagnons de galère », certains présents depuis la première heure, j'adresse également mes remerciements. Que d'heures passées ensemble, très souvent dans la bonne humeur. Je retiendrai cet esprit d'équipe, d'entre aide, chacun étant prêt à aider l'autre. Merci.

Merci à toi Fernando. Cela a été un réel plaisir de travailler avec toi sur ce mémoire. Ton goût pour la technique, ton âme de bâtisseur et ta motivation furent une des clés de la réussite de ce mémoire. Merci également à Caroline, ta femme, qui par ses petits plats et ses délicates attentions rendait les soirées de travail agréables et chaleureuses.

Enfin, je remercie ma famille, mes proches, collègues et amis pour qui j'ai été beaucoup moins disponible et présent mais qui toujours m'ont soutenu.

Laurent.

Je tiens à remercier Laurent et l'équipe du Prof. Colet pour m'avoir proposé un sujet de mémoire aussi passionnant.

Mes remerciements vont aussi au Prof. Englebert pour ses conseils qui dès le départ nous orientaient vers les bonnes solutions (peu à peu nous avons appris à déchiffrer son langage d'avant-garde)

Enfin je remercie ma famille pour la patience et les efforts dont elle a fait preuve pendant ces années d'études.

Fernando

Glossaire

Acide Aminé : constituant élémentaire des protéines. Il y en a 20 différents
Acide DésoxyriboNucléique (ADN) : Molécule biologique supportant l'information génétique grâce à l'ordre d'enchaînement des nucléotides qui le compose
Acide nucléique : Terme général désignant soit l'ARN soit l'ADN
Acide RiboNucléique (ARN) : Molécule intermédiaire supportant l'information nécessaire à la synthèse des protéines
ADN : voir Acide Désoxyribonucléique
Alignement :Superposition de séquences différentes à des fins de comparaison
AnalysisInstance : Terme issu de la norme BSA définissant un objet chargé d'invoquer un outil d'analyse BSA
AnalysisService : Terme issu de la norme BSA. Est la représentation logique d'un outil d'analyse BSA
AnalysisType : Terme issu de la norme BSA. Permet de définir les services offerts par un AnalysisService
ARN : voir Acide Ribonucléique
BSA : Biomolecular Sequence Analysis. Norme définie par l'OMG en vue de faciliter l'utilisation des logiciels d'analyses de séquences biomoléculaires
Chromosome : Support cellulaire de l'hérédité. Composé d'une double hélice d'ADN
Code génétique : Code permettant, à partir de la séquence des acides nucléiques, de déterminer la séquence des protéines
CORBA : Norme publiée par l'OMG spécifiant un modèle objet permettant l'interopérabilité de composants et d'applications réparties
Gène : Entité d'information génétique, portion d'ADN codant pour une protéine et les éléments de contrôle déterminant sa synthèse
IDL : Interface Definition Language. Langage de description de services orientés objet pour l'environnement CORBA
IOR : Interoperable Object Reference. Terme de la norme CORBA définissant une représentation standardisée des références d'objets
MDR : Master Definition Repository. Nom défini dans ce travail pour nommer la base de données contenant les définitions standardisées des AnalysisServices
Nucléotide : Constituant fondamental des acides nucléiques. Il en existe 4 différents.
OMG : Object Management Group. Consortium international définissant des standards dans le domaine des technologies orientées objets
Parser : Outil informatique permettant d'effectuer une analyse syntaxique (dans ce travail, de récupérer la valeur d'un couple champs-valeur)
Protéines : Sont les principaux acteurs du métabolisme d'un organisme formées par un enchaînement, une séquence d'acides aminés
Séquence : Pour l'ADN, chaîne de caractères basée sur un alphabet de quatre caractères (les nucléotides) dont la combinaison donne sa spécificité (séquence nucléotidique). Terme également utilisé pour définir l'ordre des acides aminés composant une protéine (séquence protéique)
Séquençage : Technique permettant la lecture de l'enchaînement des caractères sur la séquence (ADN ou protéine)

Introduction

Les recherches dans les domaines de la médecine et de la biologie font actuellement très souvent appel à la bioinformatique. Ainsi par exemple, il y a environ un an a été annoncé l'achèvement du séquençage du génome humain. Maintenant connue la succession des quatre bases A, T, C et G, pour maîtriser le sens et les subtilités de l'ADN, il faut en comprendre la syntaxe et la ponctuation. [Brown 00]. Cet objectif ambitieux ne peut être atteint sans la collaboration étroite de la recherche expérimentale et de l'analyse bioinformatique.

On peut schématiquement définir la bioinformatique comme étant l'utilisation des technologies de l'information appliquées à l'étude des données biologiques. Son domaine d'investigation comprend entre autre la fouille des données, l'alignement des séquences et la prédiction de la structure tridimensionnelle des protéines,... Les analyses sont rendues complexes par la diversité des formats des données et la multiplicité des sémantiques. De plus, la bioinformatique opère dans un environnement très hétérogène : les nombreux programmes, utilisent des données de types, de concepts, d'organisations et de formats différents. A cela s'ajoute l'hétérogénéité du hardware utilisé. Il n'est pas envisageable actuellement de réécrire l'importante collection existante de programmes relatifs à la biologie moléculaire. Il faut donc conserver ces applications patrimoines (« *legacy systems* »). Il est maintenant important d'accroître la productivité de la recherche biomédicale en favorisant l'intégration, l'interopérabilité et la réutilisation des différentes ressources (logiciels et bases de données).

CORBA (*Common Objet Request Broker Architecture*) est une plate-forme à base de technologies orientées objet. Cette norme défini un ensemble de standards permettant l'utilisation d'applications distribuées et hétérogènes. Parmi les standards, le langage de définition d'interfaces (**IDL**) permet de spécifier des données et des services tout en les rendant accessibles aux applications clientes. CORBA a été défini par *l'Object Management Group* qui a également défini en juin 2001 la norme *Biomolecular Sequence Analysis* (BSA) dans le but d'assurer l'interopérabilité des logiciels d'analyse des séquences biomoléculaires.

L'objectif de ce travail est de favoriser l'utilisation des diverses ressources bioinformatiques, et ce, sur base de l'utilisation de la technologie CORBA et de la norme BSA. Nous avons dès lors étudié ces différentes normes, rencontré le principal fournisseur de services bioinformatiques belge à la communauté scientifique (le BEN), élicité les besoins fondamentaux et proposé une architecture de fonctionnement répondant à ces normes et exigences d'où l'intitulé du mémoire : « Conception d'un système fédéré dynamique d'utilisation de ressources bioinformatiques hétérogènes »

Les chapitres successifs de ce mémoire présentent une vue d'ensemble de la bioinformatique, des applications distribuées et de CORBA. Ensuite après une étude de la norme BSA, nous détaillons l'analyse qui nous a mené à l'architecture proposée. Enfin, nous présentons la méthodologie suivie pour réaliser un prototype de notre architecture et terminons par une discussion situant ce travail parmi les diverses tentatives publiées de distribution des ressources bioinformatiques.

Chapitre 1

La bioinformatique

Après une brève introduction relative à la biologie et nécessaire à la compréhension des enjeux de ce travail, nous définirons la bioinformatique et nous détaillerons son fonctionnement actuel.

1.1 Un peu de biologie

Chez tous les êtres vivants connus sur Terre, l'information génétique - les « plans » de tout organisme - est inscrite dans l'**ADN** (pour **Acide DésoxyriboNucléique**). L'ADN est un enchaînement linéaire de quatre molécules différentes appelées **nucléotides**. Ces nucléotides sont l'adénine, la guanine, la cytosine et la thymine symbolisées par A, G, C et T et forment un code à quatre lettres. L'ordre des nucléotides (la **séquence**) de l'ADN constitue le plan directeur permettant à un organisme de se former et de fonctionner. La séquence des nucléotides est donc différente d'un organisme à un autre.

Chez l'homme, l'ADN est long de 3 milliards de nucléotides, organisés en **chromosomes** (Figure 1). L'ADN est donc la forme de stockage de l'information génétique et constitue le matériel héréditaire qui est transmis d'une génération à la suivante.

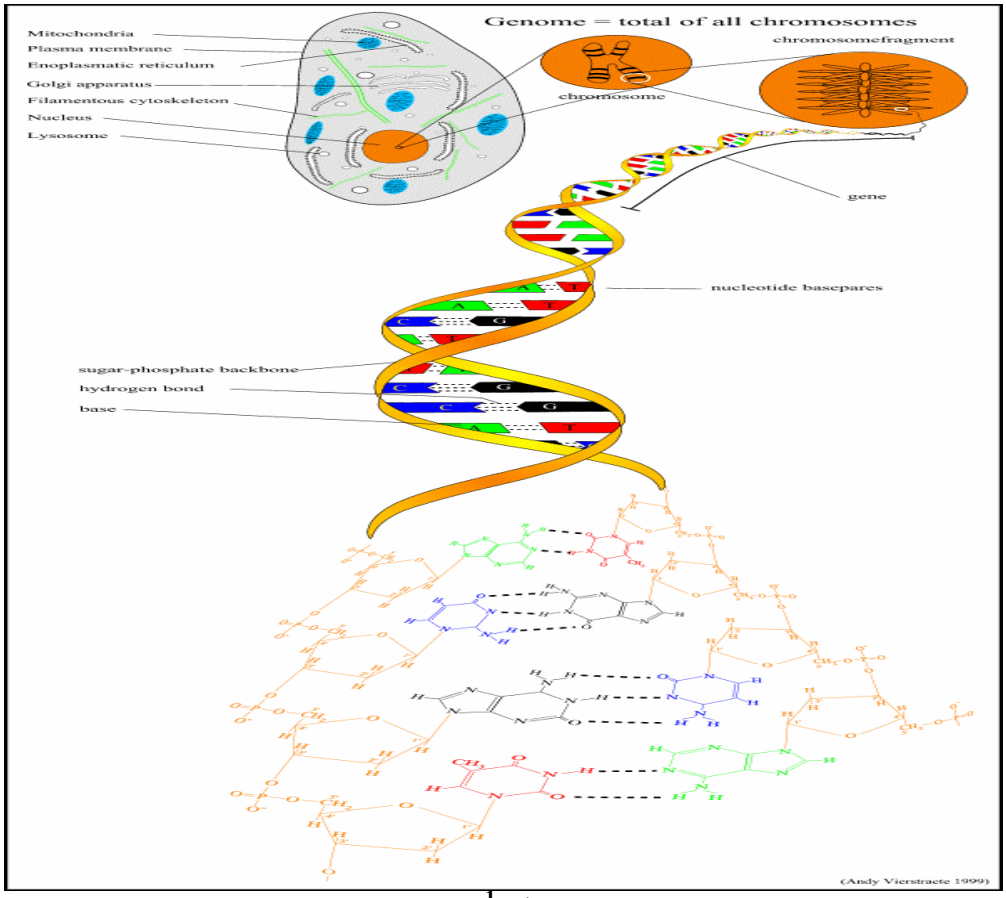

Figure 1[1] **:** l'ADN

[1] http://allserv.rug.ac.be/~avierstr/principles/cell.html

Le « dogme central » de la biologie moléculaire (Figure 2) introduit par Francis Crick (le co-découvreur de l'ADN) à la fin des années 50 stipule que, chez tous les être vivants, l'information n'est transmise que dans un sens : de l'ADN où repose l'information, à l'**ARN** - une structure transitoire permettant son transfert - vers une machine de traduction, celle ci utilisant l'information afin de synthétiser les **protéines**. Les protéines sont les constituants de base qui font fonctionner la cellule et l'organisme entier.

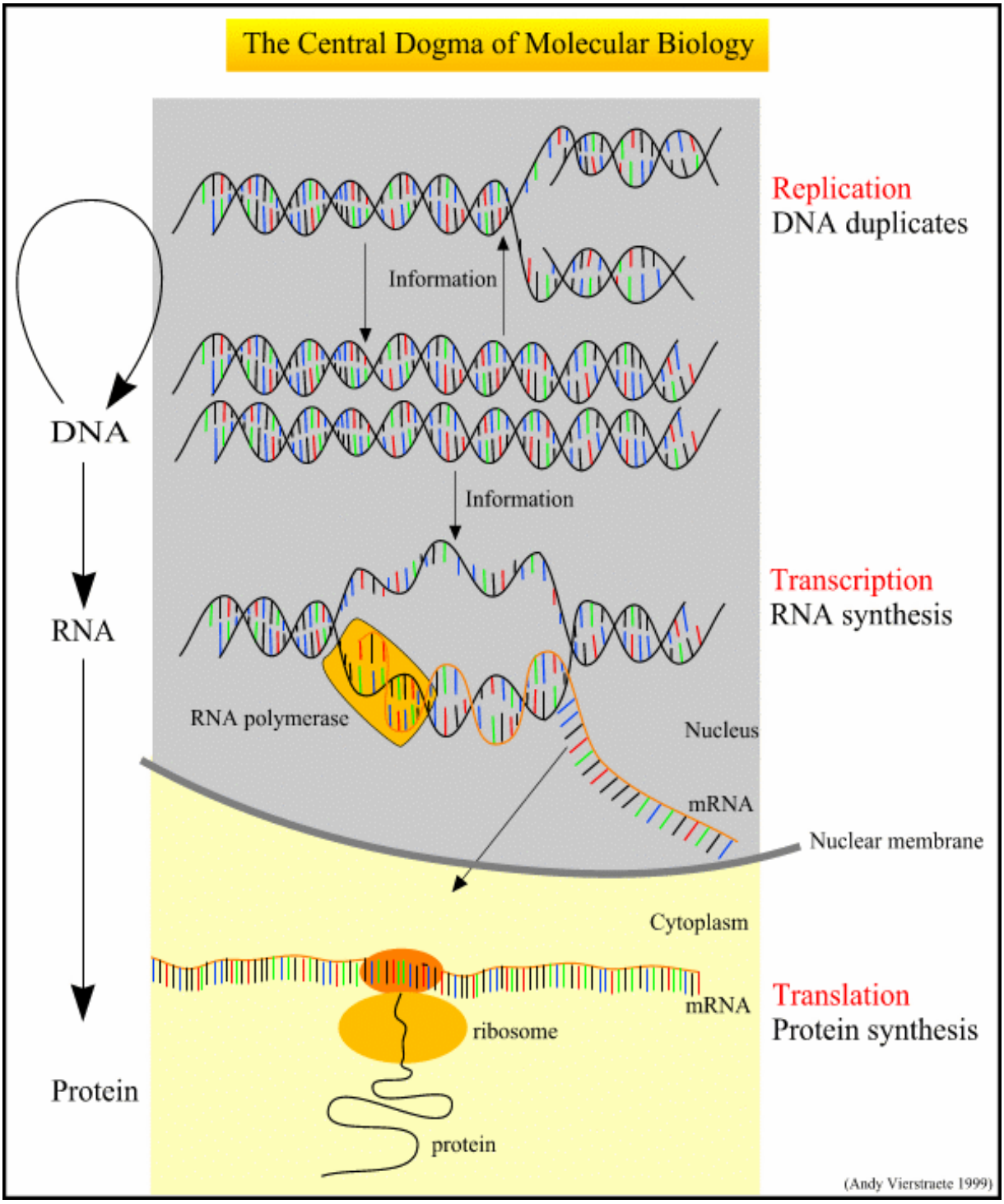

Figure 2[2] **:** Le dogme central de la biologie moléculaire

[2] http://allserv.rug.ac.be/~avierstr/principles/centraldogma.html

Une protéine est une succession d'un nombre variable de molécules, les **acides aminés** (il existe 20 acides aminés différents). La succession précise de ces acides aminés au sein d'une protéine en fournit une description précise et unique. La séquence d'une protéine est déterminée grâce au **code génétique** par la séquence d'ADN contenant l'information nécessaire à la synthèse de cette protéine. En effet, à une séquence particulière de 3 nucléotides correspond un acide aminé particulier. L'ADN codant pour la protéine X est donc différent de l'ADN codant pour la protéine Y. Chez certains organismes, tout l'ADN ne code pas toujours pour une ou des protéines. Une partie de l'ADN (ou de l'ARN de certains virus pour être exact) codant pour une protéine est appelée **gène**. L'ADN ne codant pas pour une protéine (cela représente plus de 95% des l'ADN des mammifères) a entre autre une fonction régulatrice dans la synthèse des protéines. Le **génome** d'un organisme est l'ensemble de son matériel génétique (son ADN ou son ARN pour certains virus).

Ces derniers mois, un grand pas a été accompli dans la connaissance du génome : le projet de séquençage du génome humain. La totalité de l'ADN humain a ainsi été décodé, séquencé. En effet, à l'aide de méthodes chimiques, on peut maintenant déterminer l'ordre des nucléotides composant l'ADN (cf. la Figure 5 pour un exemple de séquence obtenue). L'information récoltée est stockée dans des « banques » d'ADN. Mais obtenir la séquence d'un organisme n'est qu'un début. On dispose des « mots » du génome, il nous faut maintenant découvrir la grammaire et la syntaxe du message génétique. Quelles parties de l'ADN contrôlent les différents processus chimiques permettant la vie ? Quelles sont les parties de l'ADN codant pour une protéine et quelles sont celles ayant une fonction régulatrice ? Il faut découvrir les gènes, comprendre la régulation de leur expression (par exemple, l'hémoglobine est produite uniquement par les erythroblastes (les cellules qui vont devenir les globules rouges) et pas dans les autres types cellulaires), etc.

La quantité et le type de données dans le domaine de la biologie ne cesse d'augmenter avec, par exemple, le séquençage des génomes de différents organismes, l'analyse de la structure tridimensionnelle des protéines, etc. Toutes ces données de nature très variée sont regroupées dans des banques de données en pleine expansion. La consultation et l'analyse de ces données ne peut actuellement se faire sans l'aide de l'informatique. En effet, le challenge actuel n'est plus la récolte des données, mais bien leur analyse et leur interprétation.
C'est ici que la bioinformatique entre en jeu.

1.2 Définition de la bioinformatique

Une définition très générale de la bioinformatique est **l'utilisation des technologies de l'information au traitement des données biologiques** [Gibas 01].

Cette discipline est née de la convergence de deux révolutions technologiques : le développement des biotechnologies d'une part et des technologies de l'information d'autre part. Elle a fait son apparition dans les années 1980 avec les premières banques d'ADN centralisées (EMBL et GenBank). Il est amusant de constater que la taille de GenBank et la vitesse des processeurs ont eu la même croissance exponentielle depuis 20 ans. La figure 3 illustre la croissance de GenBank. La bioinformatique est donc une science jeune s'appuyant sur les acquis de la biologie, des mathématiques et de l'informatique

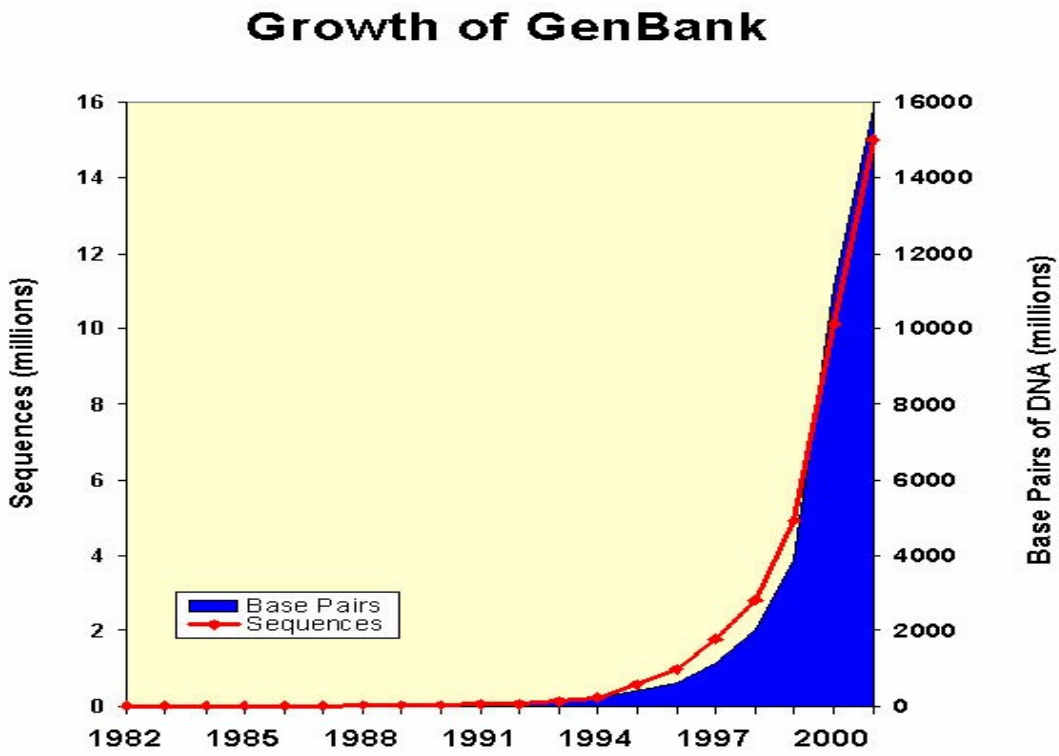

Figure 3 : Croissance de GenBank[3]

La bioinformatique propose des méthodes et des logiciels qui permettent de gérer, d'organiser, de comparer, d'analyser, d'explorer l'information génétique et génomique stockée dans les bases de données afin de prédire et de produire des connaissances nouvelles ainsi qu'élaborer de nouveaux concepts.

Les axes privilégiés autour desquels se focalise la bioinformatique sont

- la formalisation de l'information génétique
- l'analyse des séquences
- l'interprétation biologique de l'information génétique
- la prédiction fonctionnelle

La bioinformatique ne se réduit pas à une boîte à outil pour gérer, manipuler, traiter et analyser des données biologiques. C'est une approche globale, capable d'enrichir le domaine fondamental de connaissances nouvelles et d'être à l'origine de concepts biologiques nouveaux.

[3] http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html

1.3 Les bases de données biologiques

Il existe une grande variété de types de données biologiques stockées dans les bases génétiques (aussi appelées banques) : séquences d'ADN, de protéines, structure tridimensionnelle des protéines, etc. Les banques de données biologiques sont d'une importance capitale dans la recherche biomédicale. Elles regroupent des données et des résultats essentiels parfois non reproduits dans la littérature scientifique et doivent les rendre publiques.

Focalisons-nous sur les principales banques d'ADN : GenBank[4] (NCBI, Bethesda, USA), EMBL[5] (European Molecular Biology Laboratory Nucleotide Sequence Database, EBI, Cambridge, UK) et DDBJ[6] (DNA Data Bank of Japan, CIB, Mishima, Japan). Ces trois banques s'échangent systématiquement leur contenu depuis 1997 et adoptent un système de conventions communes. Ces grandes banques généralistes et internationales sont alimentées par les données produites dans tous les laboratoires. En plus des séquences, on y trouve également une bibliographie, des annotations liées aux séquences traitées, etc. On retrouve ces banques sur les serveurs des organismes les gérant, mais elles sont également dupliquées sur d'autres serveurs dans le monde.

Voici un exemple d'entrée dans GenBank

```
LOCUS       BC034026              2059 bp    mRNA    linear   HTC 08-JUL-2002
DEFINITION  Homo sapiens, albumin, clone IMAGE:4734794, mRNA.
ACCESSION   BC034026
VERSION     BC034026.1  GI:21706470
KEYWORDS    HTC.
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
REFERENCE   1  (bases 1 to 2059)
  AUTHORS   Strausberg,R.
  TITLE     Direct Submission
  JOURNAL   Submitted (02-JUL-2002) National Institutes of Health, Mammalian
            Gene Collection (MGC), Cancer Genomics Office, National Cancer
            Institute, 31 Center Drive, Room 11A03, Bethesda, MD 20892-2590,
            USA
  REMARK    NIH-MGC Project URL: http://mgc.nci.nih.gov
COMMENT     Contact: MGC help desk
            Email: cgapbs-r@mail.nih.gov
            Tissue Procurement: CLONTECH
            cDNA Library Preparation: CLONTECH Laboratories, Inc.
            cDNA Library Arrayed by: The I.M.A.G.E. Consortium (LLNL)
            DNA Sequencing by: Sequencing Group at the Stanford Human Genome
            Center, Stanford University School of Medicine, Stanford, CA  94305
            Web site:       http://www-shgc.stanford.edu
            Contact:  (Dickson, Mark) mcd@paxil.stanford.edu
            Dickson, M., Schmutz, J., Grimwood, J., Rodriquez, A., and Myers,
            R. M.

            Clone distribution: MGC clone distribution information can be found
            through the I.M.A.G.E. Consortium/LLNL at: http://image.llnl.gov
            Series: IRAL Plate: 37 Row: j Column: 20
            This clone was selected for full length sequencing because it
            passed the following selection criteria: Similarity but not
            identity to protein
            This clone has the following problem: frame shifted.

FEATURES             Location/Qualifiers
     source          1..2059
                     /organism="Homo sapiens"
                     /db_xref="taxon:9606"
                     /clone="IMAGE:4734794"
                     /tissue_type="Liver"
                     /clone_lib="NIH_MGC_76"
                     /lab_host="DH10B"
                     /note="Vector: pDNR-LIB"
```

```
BASE COUNT     644 a    411 c    446 g    558 t
ORIGIN
        1 agcttttctc ttctgtcaac cccacacgcc tttggcacaa tgaagtgggt aacctttatt
       61 tcccttcttt ttctctttag ctcggcttat tccaggggtg tgtttcgtcg agatgcacac
      121 aagagtgagg ttgctcatcg gtttaaagat ttgggagaag aaaatttcaa agccttggtg
      181 ttgattgcct ttgctcagta tcttcagcag tgtccatttg aagatcatgt aaaattagtg
      241 aatgaagtaa ctgaatttgc aaaaacatgt gttgctgatg agtcagctga aaattgtgac
      301 aaatcacttc ataccctttt tggagacaat tatgcacagt tgcaactctt cgtgaaacct
      361 atggtgaaat ggctgactgc tgtgcaaaac aagaacctga gagaaatgaa tgcttcttgc
      421 aacacaaaga tgacaaccca aacctcccc gattggtgag accagaggtt gatgtgatgt
      481 gcactgcttt tcatgacaat gaagagacat ttttgaaaaa atacttatat gaaattgcca
      541 gaagacatcc ttacttttat gccccggaac tccttttctt tgctaaaagg tataaagctg
      601 cttttacaga atgttgccaa gctgctgata aagctgcctg cctgttgcca aagctcgatg
      661 aacttcggga tgaagggaag gcttcgtctg ccaaacagag actcaagtgt gccagtctcc
      721 aaaaatttgg agaaagagct ttcaaagcat gggcagtagc tcgcctgagc cagagatttc
      781 ccaaagctga gtttgcagaa gtttccaagt tagtgacaga tcttaccaaa gtccacacgg
      841 aatgctgcca tggagatctg cttgaatgtg ctgatgacag ggcggacctt gccaagtata
      901 tctgtgaaaa tcaagattcg atctccagta aactgaagga atgctgtgaa aaacctctgt
      961 tggaaaaatc ccactgcatt gccgaagtgg aaaatgatga gatgcctgct gacttgcctt
     1021 cattagctgc tgattttgtt gaaagtaagg atgtttgcaa aaactatgct gaggcaaagg
     1081 atgtcttcct gggcatgttt ttgtatgaat atgcaagaag gcatcctgat tactctgtcg
     1141 tgctgctgct gagacttgcc aagacatatg aaaccactct agagaagtgc tgtgccgctg
     1201 cagatcctca tgaatgctat gccaaagtgt tcgatgaatt taaacctctt gtggaagagc
     1261 ctcagaattt aatcaaacaa aattgtgagc tttttgagca gcttggagag tacaaattcc
     1321 agaatgcgct attagttcgt tacaccaaga aagtacccca agtgtcaact ccaactcttg
     1381 tagaggtctc aagaaaccta ggaaaagtgg gcagcaaatg ttgtaaacat cttgaagcaa
     1441 aaagaatgcc ctgtgcagaa gactatctat ccgtggtcct gaaccagtta tgtgtgttgc
     1501 atgagaaaac gccagtaagt gacagagtca ccaaatgctg cacagaatcc ttggtgaaca
     1561 ggcgaccatg cttttcagct ctggaagtcg atgaaacata cgttcccaaa gagtttaatg
     1621 ctgaaacatt caccttccat gcagatatat gcacactttc tgagaaggag agacaaatca
     1681 agaaacaaac tgcacttgtt gagcttgtga aacacaagcc caaggcaaca aaagagcaac
     1741 tgaaagctgt tatggatgat ttcgcagctt ttgtagagaa gtgctgcaag gctgacgata
     1801 aggagacctg ctttgccgag gagggtaaaa aacttgttgc tgcaagtcaa gctgccttag
     1861 gcttataaca tcacatttaa aagcatctca gcctaccatg agaataagag aaagaaaatg
     1921 aagatcaaaa gcttattcat ctgtttttct ttttcgttgg tgtaaagcca acaccctgtc
     1981 taaaaaacat aaatttcttt aatcattttg cctcttttct ctgtgcttca aaaaaaaaaa
     2041 aaaaaaaaaa aaaaaaaaa
//
```

[4] http://www.ncbi.nih.gov/Genbank/GenbankOverview.html
[5] http://www.ebi.ac.uk/embl/
[6] http://www.ddbj.nig.ac.jp/

GenBank [Anonyme 01] est un ensemble de fichiers de données « à plat » (flat file) où les données sont codées en ASCII. A l'intérieur de ces fichiers, chaque séquence est contenue dans une structure appelée « entrée », une entrée comprenant une quantité variable d'informations liées à la séquence considérée (l'organisme dont elle est issue, son rôle, etc.) structurées selon une convention champ/valeur. Ainsi, chaque entrée de la base est composée de lignes qui commencent par mot clé. Par exemple ACCESSION pour le numéro d'accession unique, REFERENCE pour les références des articles décrivant la séquence et ORIGIN pour la séquence proprement dite. Chaque entrée se termine par « // ».
Notons, et c'est très important, que ce format n'est pas standardisé et diffère selon les bases de données (ce qui implique par exemple l'usage de **parsers** différents).

Concernant la diffusion des données, fort heureusement nous n'en sommes plus à l'époque (1982) où les données reprises dans GenBank par exemple étaient distribuées sur bandes magnétiques et ce, quatre fois par an. Actuellement, ces mises à jours se font via Internet et sont journalières. Les principales bases de données biologiques (et les outils de recherche) ainsi que les logiciels d'analyse utilisant des ordinateurs parmi les plus puissants sont maintenant disponibles via Internet.

La difficulté dans la gestion des données biologiques provient non pas de leur quantité (il y a par exemple actuellement plus de quinze millions d'entrées dans GenBank représentant 53 GigaBytes) mais bien de leur complexité [Achard 01].

- Les données sont complexes à modéliser, il en existe de nombreux types différents avec de nombreuses relations
- De nouveaux types de données émergent régulièrement (séquence de génomes complets, microarrays, etc.). La modélisation de ces données crée de nouvelles relations avec les données existantes et peut entraîner une modification de notre perception de l'ensemble
- L'analyse des données en génère de nouvelles devant être modélisées et intégrées
- Les données sont fréquemment mises à jour, accédées et échangées de manière très intensive via Internet par les chercheurs

Techniquement, il est important de rappeler que le volume des données biologiques double en moins de deux ans, qu'elles sont disséminées dans de nombreuses bases de données souvent dupliquées en différentes localisations, et que les formats de données sont très hétérogènes. Un des challenges actuels de la bioinformatique est donc l'intégration de ces ressources nombreuses, complexes et hétérogènes.

1.4 L'interrogation des bases de données

L'accès aux bases de données nécessite l'intervention de programmes capables de lire et d'extraire l'information voulue des bases selon des critères choisis. Le réseau Internet et le développement d'interfaces de programmes situés sur des serveurs distants ont complètement modifié la manière d'échanger et de chercher l'information ainsi que les méthodes de travail. En effet, le langage HTML a permis de définir des interfaces simples (les scripts CGI pour *Common Gateway Interface*) pour des programmes lancés sur un serveur. Parmi ces programmes, citons simplement SRS (*Sequence Retrieval System*)[7] qui permet d'interroger à

[7] http://srs.ebi.ac.uk/

l'aide d'une même interface pratiquement n'importe quelle collection de séquences disponibles sous forme de fichiers texte [Zdobnov 02]. Ce programme contient plusieurs parsers capables d'extraire l'information des différentes banques et de présenter, reformater l'information de manière plus uniforme (mais toujours dans un format « à plat »). SRS est proposé par de nombreux sites serveurs.

1.5 Les logiciels d'analyse

De nombreux algorithmes ont été développés afin d'étudier les bioséquences. Ces algorithmes ont été intégrés dans des logiciels, souvent regroupés au sein de « suites » offrant une interface, un mode d'utilisation et des types de données standardisés au sein de la suite (par exemple l' *European Molecular Biology Open Software Suite*[8]). Il existe fréquemment des interfaces web permettant leur accès via Internet. Ces logiciels et les différentes bases de données génétiques sont alors regroupés sur des serveurs offrant un « portail » pour ces services. Par exemple en Belgique, le *Belgian EMBnet Node* (BEN)[9] qui fait partie du réseau européen EMBnet[10] fourni au monde académique et industriel un accès aux bases de données génétiques et aux outils d'analyses de ces données.

1.6 Présentation du fil rouge

Lors de ce travail, nous prendrons un exemple qui nous servira à illustrer notre propos où un biologiste veut comparer les gènes de l'albumine humaine et de souris. Pour ce faire, il doit rechercher les séquences correspondantes dans les banques d'ADN, isoler dans les entrées trouvées l'information concernant la séquence et aligner (comparer) les deux séquences afin d'observer les éventuelles différences entre les deux séquences.

Les captures d'écran ci dessous illustrent les différentes étapes à effectuer actuellement afin de faire cette comparaison de séquences. La première étape (Figure 4) consiste à se connecter à un serveur web donnant accès au logiciel SRS, et faire une recherche dans GenBank des séquences de l'albumine humaine.

[8] http://www.hgmp.mrc.ac.uk/Software/EMBOSS/
[9] http://ben.vub.ac.be/
[10] http://www.embnet.org/

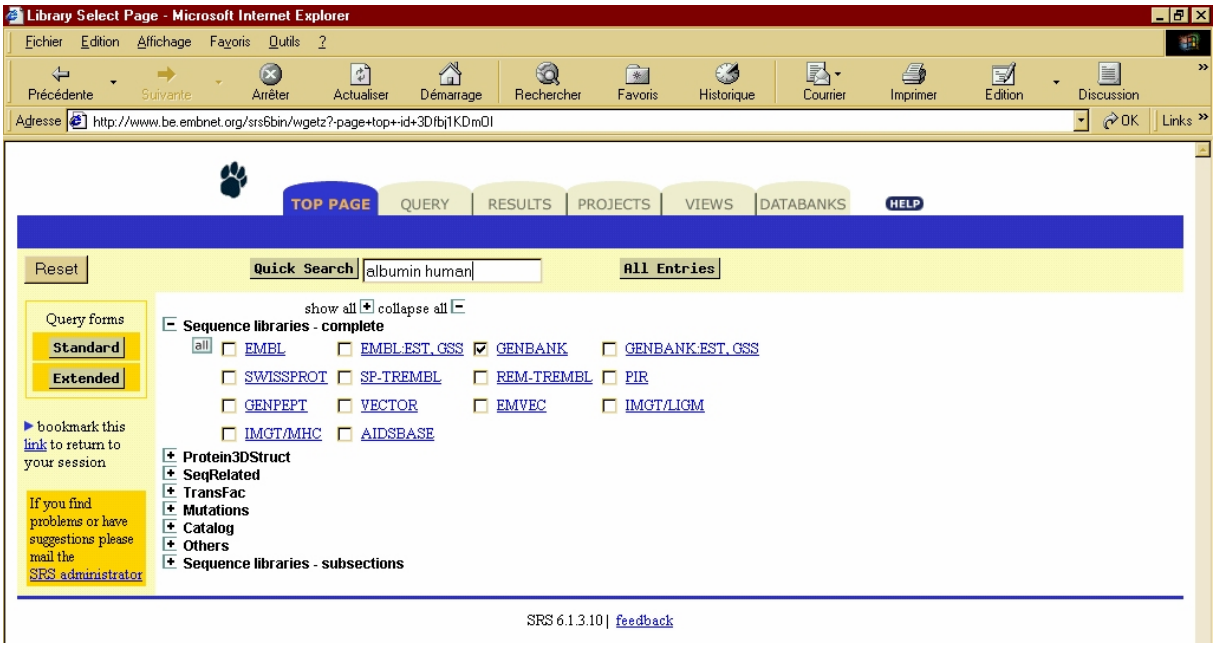

Figure 4 : Exemple de recherche à l'aide de SRS

Parmi les entrées renvoyées, le biologiste en sélectionnera une. (figure 5)

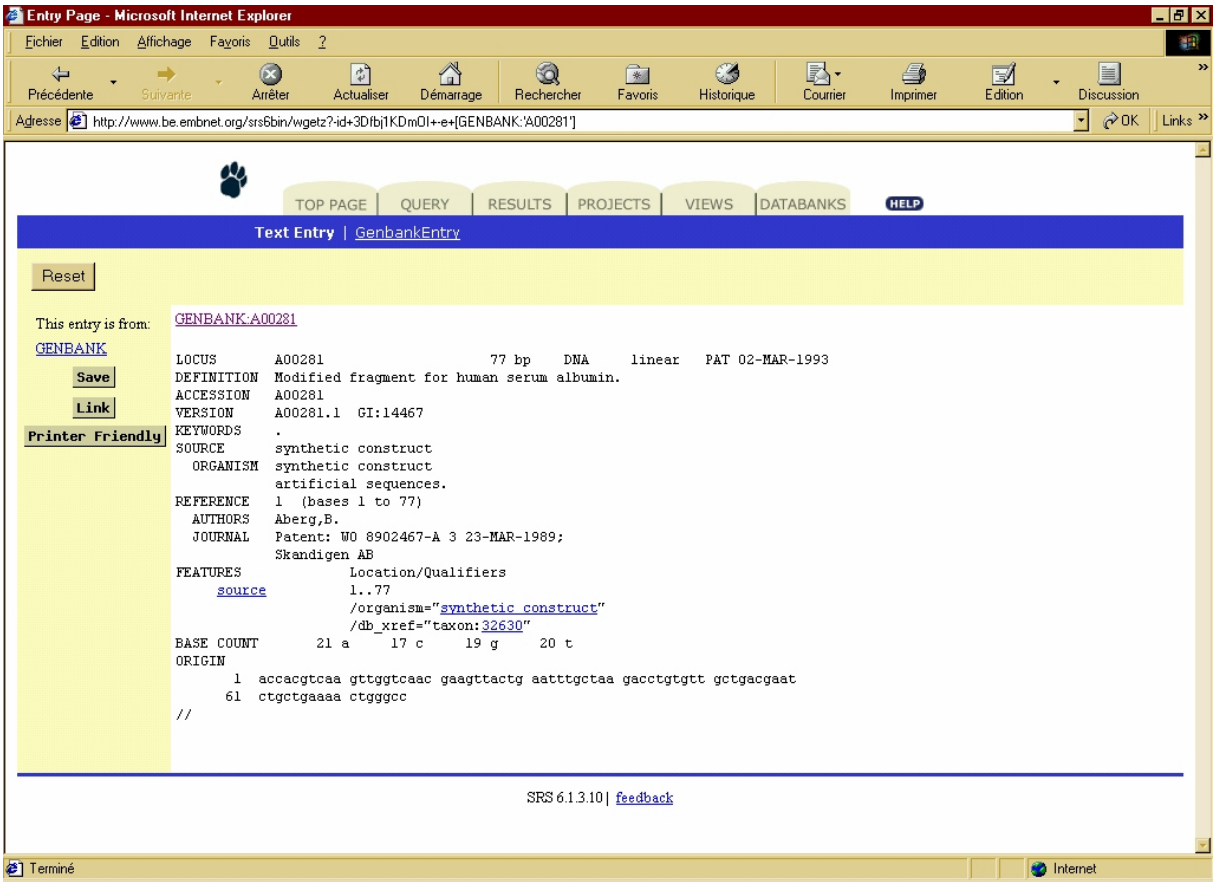

Figure 5 : Récupération d'une entrée avec SRS

Puis il en isole la séquence (Figure 6) au format requis par le logiciel qu'il utilisera pour l'**alignement**

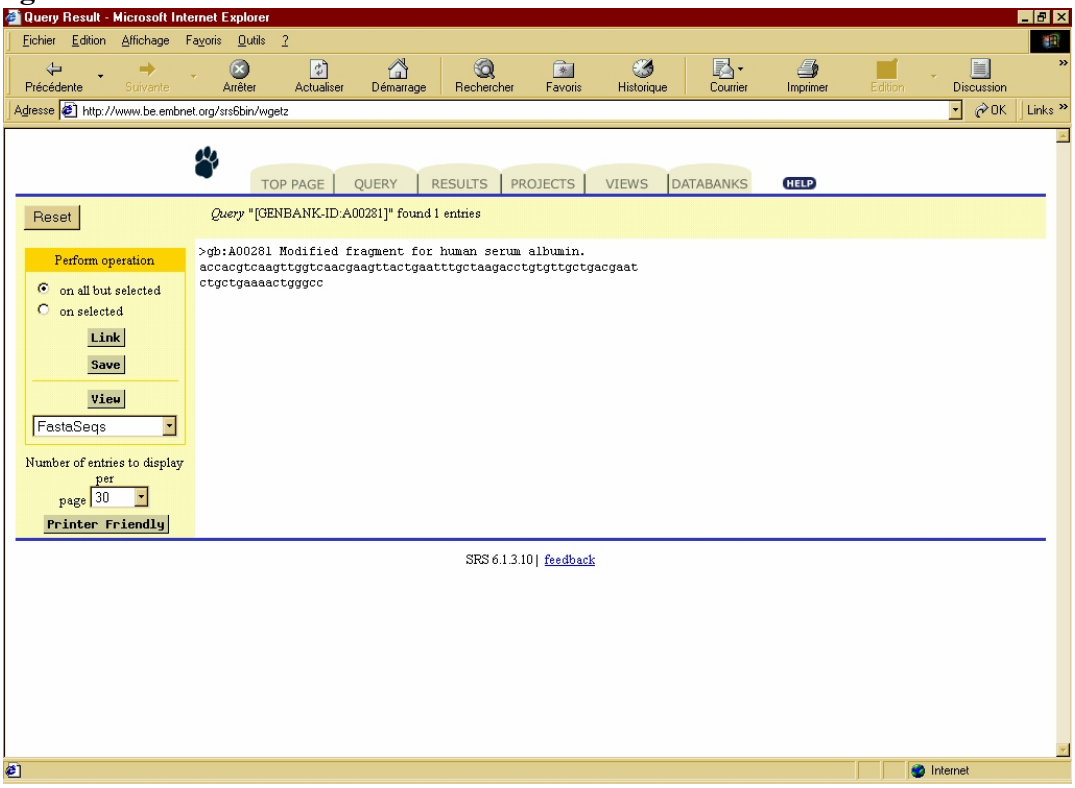

Figure 6 : Isolement de la séquence dans un format spécifique

Après avoir fait la même chose pour l'albumine de souris, il faut se connecter sur un serveur offrant l'outil d'analyse recherché (figure 7).

Figure 7 : Menu général de la suite EMBOSS

Après avoir sélectionné le logiciel voulu et introduit les séquences à comparer, l'alignement sera renvoyé :

```
########################################
# Program:  matcher
# Rundate:  Fri Dec 13 14:41:29 2002
# Report_file: mmu011413.matcher
########################################
#=======================================
#
# Aligned_sequences: 2
# 1: MMU011413
# 2: A00281
# Matrix: EDNAFULL
# Gap_penalty: 16
# Extend_penalty: 4
#
# Length: 77
# Identity:      56/77 (72.7%)
# Similarity:    56/77 (72.7%)
# Gaps:           0/77 ( 0.0%)
# Score: 196
#
#
#=======================================


       220       230       240       250       260
MMU011 AGCATGCCAAATTAGTGCAGGAAGTAACAGACTTTGCAAAGACGTGTGTT
       : :: : ::: :: ::  : ::::: :: :: ::::: ::::: ::::::
A00281 ACCACGTCAAGTTGGTCAACGAAGTTACTGAATTTGCTAAGACCTGTGTT
               10        20        30        40        50

       270       280       290
MMU011 GCCGATGAGTCTGCCGCCAACTGTGAC
       :: :: :: ::::: :  ::::: : :
A00281 GCTGACGAATCTGCTGAAAACTGGGCC
               60        70

#---------------------------------------
#---------------------------------------
```

Cet alignement sera interprété par le biologiste.

Chapitre 2

L'informatique distribuée

Ce chapitre abordera quelques caractéristiques de l'informatique distribuée. Ensuite, nous verrons comment l'Object Management Group tente de faciliter son utilisation à travers l'écriture de différentes normes, comme CORBA.

2.1 Les environnements répartis

Le développement du matériel informatique et des technologies réseaux favorise considérablement l'évolution des systèmes d'information grâce à la mise en place d'architectures distribuées. Ces applications peuvent être très réparties et tourner sur différentes machines sous des environnements hétérogènes.

Les principaux attraits d'une architecture répartie apparaissent rapidement :

- **Amélioration des performances**
  La division d'un application en plusieurs unités permet d'uniformiser la charge de travail de manière statique (lors de la phase de conception) ou dynamique (lors de la phase d'exécution). Chaque unité peut-être une cause d'échec mais n'entraînera pas nécessairement l'indisponibilité du système.

- **Fiabilité**
  La multiplication d'unités de fonctionnement va accroître la disponibilité de l'application en cas de défaillance d'une unité.

La conception de telles applications est cependant relativement complexe, elle doit prendre en compte les critères de communication entre applications, les problèmes d'intégration de l'existant , les problèmes d'hétérogénéité et d'interopérabilité.

- **La communication**

  Une application distribuée se compose de plusieurs éléments logiciels et matériels tels que logiciels clients, serveurs, machines, espace de stockage de données, réseaux et protocoles de communication entre machines et logiciels. La communication entre ces différents éléments doit s'effectuer grâce à une infrastructure de communication entre les machines distribuées, c'est-à-dire à travers le réseau. Cette infrastructure doit permettre de connecter plusieurs machines à travers différents réseaux physiques.

- **L'hétérogénéité**

  Dans un environnement distribué, il est nécessaire de prendre en compte le grand nombre de solutions technologiques pouvant cohabiter. Les types de matériels peuvent être très variés (simple ordinateur ou supercalculateur) et les systèmes d'exploitation peuvent être différents.

- **L'intégration**

Il est nécessaire de minimiser les problèmes liés à l'hétérogénéité en intégrant ces différentes technologies pour rendre l'application globalement cohérente et fonctionnelle. Par ailleurs, même si de nouvelles technologies peuvent être intégrées, il est souvent indispensable de préserver les données « patrimoines » des entreprises (« legacy »). L'approche consiste alors à conserver les données et services passés tout en offrant de nouveaux services apportés notamment par des techniques récentes (Web …)

- **L'interopérabilité**

  Une solution propriétaire qui a pris en compte les trois concepts ci-dessus peut encore devoir communiquer avec des applications développées par des tiers. Il est alors impératif de définir des normes suivies par tout les acteurs soucieux de coopérer.

2.2 L'utilisation des plates-formes réparties

Pour mettre en œuvre les différents critères énoncés ci-dessus, il est utile de posséder des mécanismes généraux permettant l'utilisation d'une application répartie . Il faut par exemple retrouver les ressources partagées sur le réseau, réaliser des transactions entre différentes applications, assurer la persistance des ressources partagées ou encore assurer la sécurité des communications.

En vue d'éviter l'étude de ces différents mécanismes à chaque nouvelle application des plates-formes réparties – ou **middlewares** - ont été élaborées. Ces middlewares proposent des services performants qui peuvent être directement intégrés dans les applications et rendent transparents diverses caractéristiques :

- Transparence des accès
  L'accès à un composant local ou distant est identique

- Transparence de la localisation
  On peut accéder à un composant sans devoir connaître son adresse physique.

- Transparence de la migration
  Une panne ou un déménagement d'ordinateur n'affecte pas l'utilisation.

- Transparence de la réplication
  L'application reste cohérente même en cas de réplication des données.

- Transparence de la concurrence
  L'utilisateur ne perçoit pas la concurrence entre différents processus.

- Transparence de l'évolution
  L'application supporte l'ajout de nouveaux composants.

Les middlewares les plus utilisés sont sans doute DCOM et **CORBA** (*Common Object Request Broker Architecture*)[11]**.** CORBA permet d'envoyer des requêtes d'un client vers un

[11]http://www.corba.org

serveur à travers un bus logiciel. En vue de garantir la portabilité du code et l'interopérabilité de l'application, les middlewares doivent absolument être standardisés.

2.3 L'Object Management Group

L'**Object Management Group (OMG)**[12] est un consortium international à but non lucratif créé en 1989. Il est composé de plus de 800 membres provenant de fournisseurs de systèmes ou de logiciels ainsi que des utilisateurs finaux de l'industrie informatique, et des chercheurs. Son objectif est de produire et de maintenir des spécifications afin de favoriser l'intégration d'applications distribuées hétérogènes et la réutilisation de composants logiciels et ce via la théorie et la pratique des technologies orientées objet dans le développement logiciel.

Ainsi, la norme définissant CORBA est issue de l'OMG. CORBA est un standard décrivant une architecture permettant de faire collaborer des applications écrites dans des langages différents et disposées sur des machines, des environnements différents.

L'OMG organise la standardisation pour l'interopérabilité des logiciels par domaines de métiers, qualifiées d'intégration verticale, en suscitant la création de *Domain Task Forces* (DTF). La récente *Life Science Research DTF* [13] est consacrée aux sciences de la vie, dont la biologie moléculaire et la génomique. Ses travaux portent sur l'élaboration de représentations communes pour l'interconnexion des bases de données biologiques et des logiciels de la bioinformatique.

2.4 Présentation de CORBA

CORBA (Common Object Request Broker Architecture) est une spécification décrivant un « middleware » orienté objet. Ce bus d'objets répartis offre un support d'exécution masquant les couches techniques d'un système réparti (langage de programmation, système d'exploitation, processeur et réseau) et il prend en charge les communications entre les composants logiciels formant les applications hétérogènes. [Geib 99]

Object Model

Il s'agit d'un modèle générique assurant la communication avec des systèmes orientés objets conformes au modèle OMG.

ORB (Object Request Broker)

Il s'agit du bus à requête d'objets [Daniel 00], en charge d'assurer la communication entre les applications distribuées. Cette communication est basée sur le mécanisme d'invocation de procédures distantes. Pratiquement, (figure 8) cette communication requiert la création d'amorces au niveau client et au niveau serveur. Ces amorces permettent aux applications de se connecter au bus afin d'échanger des messages. Le client a une vue « objet » du serveur, c'est-à-dire qu'il le voit en tant que fournisseur d'objets qu'il pourra utiliser. L'ORB est donc responsable de trouver l'implémentation de l'objet d'une requête, de préparer l'implémentation nécessaire à la recevoir et communiquer ses données constitutives.

[12] http://www.omg.org
[13] http://www.omg.org/homepages/lsr/

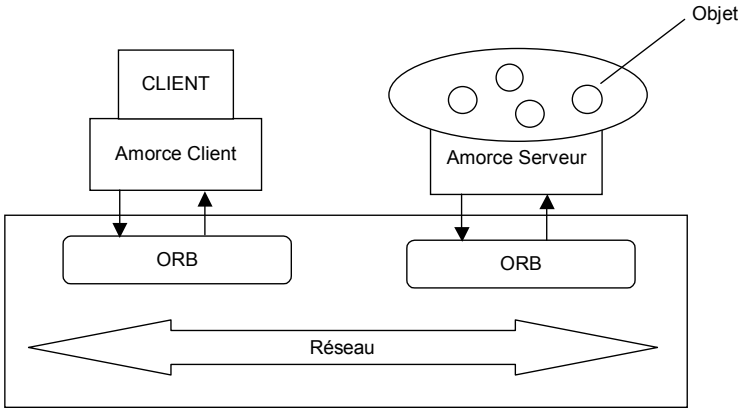

Figure 8 : Le modèle CORBA

Le langage IDL

IDL (Interface Description Language) est un langage de description d'objets distribués neutre indépendant des langages de programmation et des architectures. Avec CORBA, un objet distribué est vu comme une entité offrant un certain nombre de services accessibles à distance. L'intérêt majeur du langage IDL est de permettre la définition des contrats passés entre les fournisseurs d'objets CORBA et leurs utilisateurs. Les fournisseurs décrivent, grâce au langage IDL, l'ensemble des interfaces des objets qu'ils veulent fournir à leurs clients ; ensuite, les clients utilisent ces interfaces IDL pour exploiter à travers leurs programmes les objets ainsi fournis. Par exemple, (figure 9) si l'on imagine une application bancaire élémentaire, les services offerts par les objets distribués « Banque » et « Compte » sont décrits en IDL (ici il s'agit de la possibilité de dépôt ou de retrait d'argent).

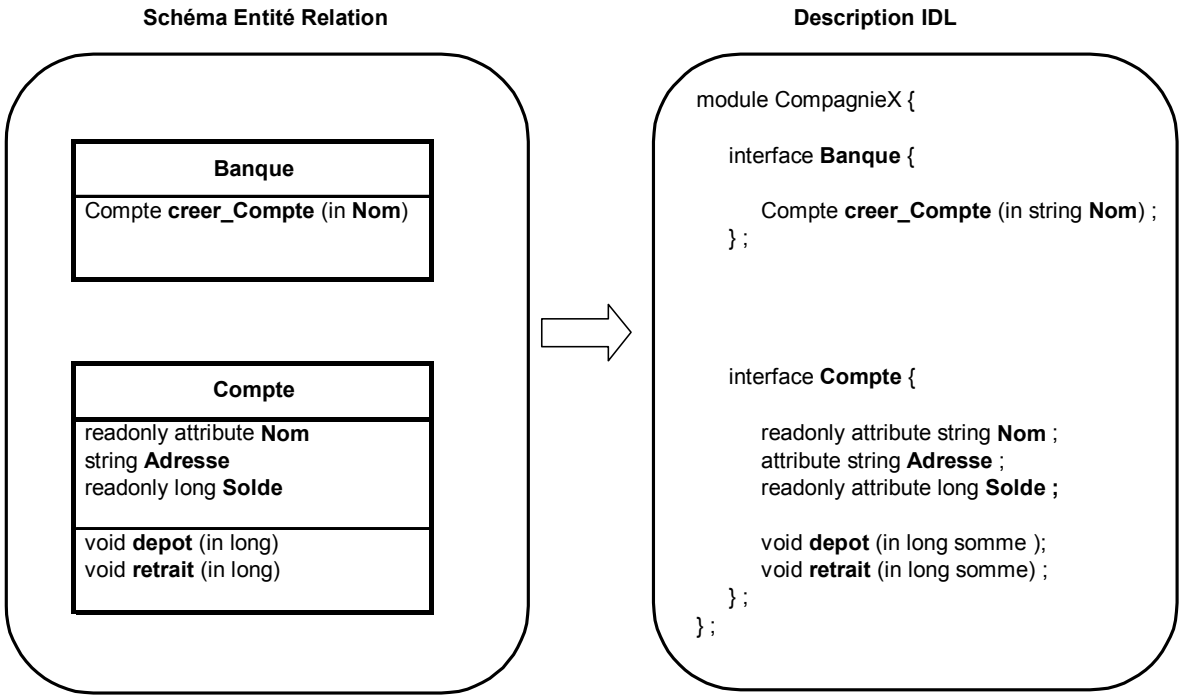

Figure 9 : Exemple d'application définie en IDL

- Un module sert à regrouper des définitions de type qui ont un intérêt commun. Cette structuration permet aussi de limiter les conflits de nom pouvant intervenir entre plusieurs spécifications.
- Le composant « interface » est apparenté aux interfaces habituellement utilisées dans les langages orientés objets.
- Le composant « valuetype » est une notion plus récente apparue dans la version CORBA 3 et n'est pas encore supporté par tout les ORBs. Il permet à un client de récupérer un objet distant par valeur.

Object Services

Il s'agit d'un ensemble de services spécifiés par des Interfaces IDL qui se branchent sur l'ORB et qui assurent des fonctionnalités système communes à la majorité des applications distribuées. Ce sont les services Cycle de vie, Nommage, Persistance, Evénements, Transactions, Concurrence d'accès, Relations, Externalisation, Interrogation, Collections, Licences, Propriétés, Sécurité, Temps, Vendeur, Changements. Les services utilisés dans notre travail sont

- Le cycle de vie *(Life Cycle Service*) qui gère la création, le déplacement et la destruction des objets CORBA.
- Le service de propriétés (*Property Service*) qui permet à un objet d'associer des informations appelées propriétés (*Properties*).
- Le service de temps (*Time Service*) qui permet la synchronisation des objets à l'échelle du réseau en fournissant une horloge globale sur le bus.
- Le service des événements (*Event Service*) qui permet aux objets de produire des événements asynchrones en utilisant un intermédiaire appelé le *canal d'événements*.

Common Facilities

Il s'agit de modèles applicatifs permettant la résolution de problèmes communs à de nombreuses architectures distribuées comme la gestion des utilisateurs, des tâches ou l'administration système.

Architecture CORBA

Rappelons que l'OMG a seulement écrit la norme spécifiant CORBA, elle ne précise aucun détail d'implémentation mais définit un ensemble de modules fonctionnels dont chacun est responsable d'une partie de la communication.

Modules principaux (figure 10)

- L'ORB Core (noyau de communication)
  Il permet de transporter les invocations entre les objets en utilisant le protocole **IIOP** (Internet Inter ORB Protocol) spécifié par l'OMG. L'ORB n'est pas directement accessible par les applications à l'exception de quelques fonctions principales.

- Static Invocation Interface – SII (Interface d'invocations statiques)
  Cette interface permet au client d'invoquer les objets distants de manière transparente. Elle a en charge d'emballer les requêtes, de déballer les résultats et de procéder à la requête.

- Dynamic Invocation Interface –DII (Interface d'invocations dynamiques)
  Il s'agit de l'interface permettant de construire des requêtes dynamiquement. L'utilisation de ce module est plus complexe car il nécessite la spécification de chaque élément de la requête (objet cible, nom de méthode, paramètres) lors de l'exécution.

- Static Skeleton Interface – SSI (Interface de squelette statique)
  Ce module est situé du côté serveur et sert à déballer les requêtes afin de les transmettre aux objets d'implantation (ce sont les objets qui sont en charge d'exécuter les requêtes – on les appelle les **servants**). Au retour des servants, les résultats sont emballés afin d'être envoyés vers le SII client.

- Dynamic Skeleton Interface – DSI (Interface de squelette dynamique)
  Il s'agit de l'équivalent du DII mais côté serveur.

- Object Adaptator – OA (Adaptateur d'objets)
  Le rôle de l'adaptateur d'objet est de gérer les objets hébergés par le serveur. C'est lui qui active un objet (le rend accessible au client) et qui le désactive. Depuis la version CORBA 2.2 (1998) l'OA spécifié est le POA (Portable Object Adaptor).

- Interface Repository IR (Référentiel d'Interfaces)
  Il permet de stocker de manière persistante les informations définies par les spécifications IDL disponibles au moment de l'exécution.

- Implementation Repository ImpR (Référentiel des Implantations)
  Ce module est spécifique à chaque fournisseur d'ORB. De manière générale, il contient l'ensemble des informations qui décrivent l'implémentation des objets comme par exemple le nom des exécutables contenant le code des objets ou les droits d'accès au serveur.

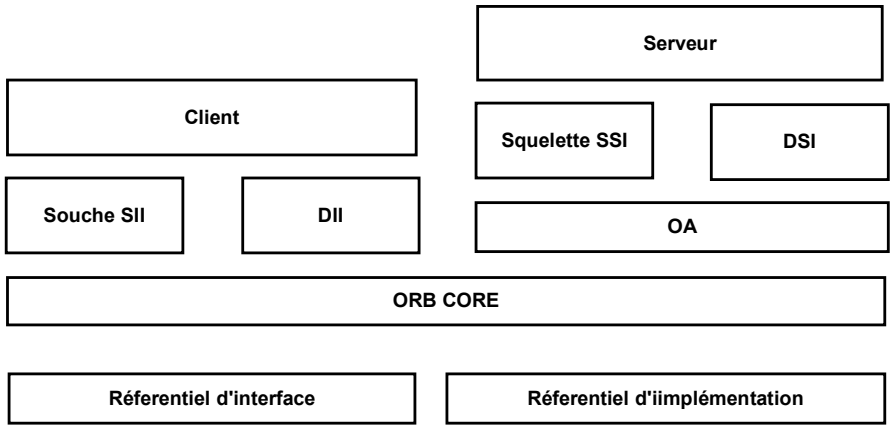

Figure 10 : Vue de l'architecture

Protocole

L'OMG a spécifié un protocole générique réseau à usage général appelé **GIOP** (General Inter-ORB Protocol). Afin d'instancier ce protocole sur la couche transport TCP/IP d'Internet, le protocole **IIOP** (Internet Inter-ORB Protocol) a été défini, mais GIOP peut également être instancié sur d'autres couches de transport comme IPX/SX fonctionnant sur les réseaux Novell.

Dans le protocole GIOP, les éléments suivant ont été spécifiés :

- la représentation commune des données offre la projection des types de données IDL en une représentation physique pouvant être véhiculée sur le réseau.
  Ce format est le **CDR** (Common Data Representation).

- Le format des messages (appel de méthode, exception)

- Les pré-requis sur la couche transport du réseau (par exemple l'ordonnancement des messages).

- Les références d'objets interopérables appelés **IOR** (Interoperable Object Reference).
  Ces références sont utilisées pour localiser et identifier un objet hébergé par un serveur. Elles contiennent le numéro de version IIOP utilisé, mais surtout l'adresse IP de la machine, le port d'écoute du serveur ainsi qu'un identifiant de l'objet au sein même du serveur.

Chapitre 3

La norme BSA

Nous allons à présent étudier la norme **BSA** et voir comment elle peut répondre à certains problèmes de la bioinformatique actuelle.

La norme BSA (Biomolecular Sequence Analysis) est une norme publiée par la « Life Science Research Task Force » ou LSR. La version 1.0 a été publiée en juin 2001 et est disponible sur le site de l'OMG [14]. La **LSR** a proposé un ensemble de spécifications dans le but d'assurer l'interopérabilité des logiciels d'analyse de séquences biomoléculaires et ce via l'utilisation de **CORBA** et son langage de description d'interface IDL.

La norme BSA contient deux modules IDL principaux qui sont le **DsLSRBioObjects** (modélisant l'aspect « données ») et le **DsLSRAnalysis** (modélisant les mécanismes d'analyses).

3.1 Module DsLSRBioObjects

Nous avons vu précédemment que les formats des données était très hétérogène dans les différentes bases de données. Le module DsLSRBioObjects décrit le format des données de séquences biomoléculaires. Ce module contient diverses interfaces (figure11) articulées autour d'une interface centrale appelée « **Interface BioSequence** ». L'objet BioSequence est une abstraction d'une séquence biologique comme une suite ordonnée de nucléotides composés par une chaîne d'ADN ou encore une suite d'acides aminés. Une BioSequence peut avoir n'importe quelle longueur (le nombre de nucléotides ou de bases la composant) et n'importe quelle signification, mais elle doit fournir les caractéristiques essentielles des séquences biologiques (nom, identification, description, longueur) et les opérations nécessaires pour retirer totalement ou partiellement sa valeur.

[14] http://www.omg.org/technology/documents/formal/biomolecular_sequence.htm

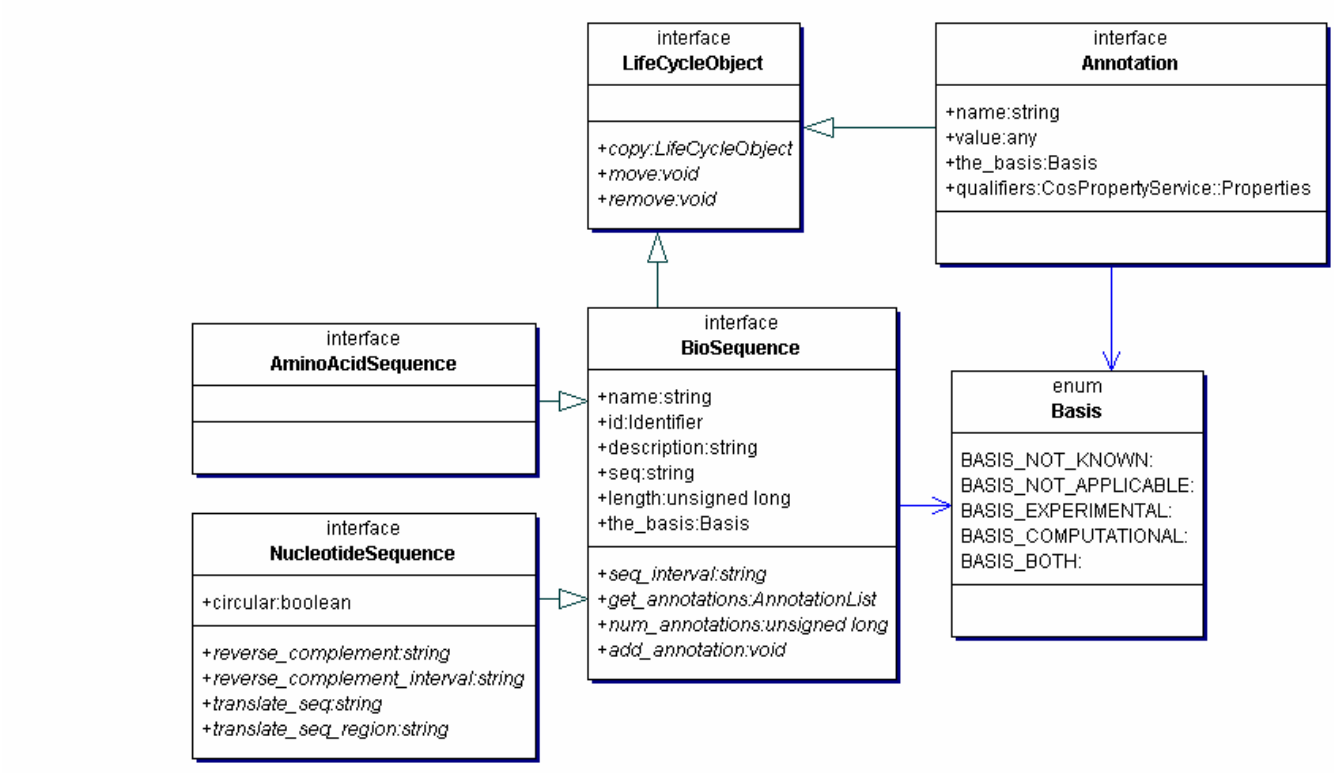

Figure 11 : Principales Interfaces du module DsLSRBioObjects

L'interface BioSequence spécialise deux sous-types qui sont l'interface « **NucleotideSequence** » et l'interface « **AminoAcidSequence** » soit respectivement séquence nucléique et séquence protéique.

- L'interface NucleotideSequence représente une séquence d'ADN ou d'ARN et fournit diverses opérations nécessaires à la manipulation des données constituant la séquence.

- L'interface AminoAcidSequence représente une protéine et ne contient aucune opération associée.

Afin de décrire les objets BioSequence, la norme a prévu de leur associer une **annotation**.
Ces annotations sont composées de paires nom-valeur. Le nom spécifie le type général de l'annotation et la valeur contient l'annotation elle-même. Le type associé à cette valeur est le type « **any** »[15].

La manière dont la séquence a été expérimentée est décrite à l'aide de l'attribut « **the_Basis** » qui indique si la séquence a été déterminée par expérimentation ou par calcul (par exemple lors d'une transcription in silico d'une séquence déterminée expérimentalement)
Ce module contient également des interfaces décrivant les alignements par exemple. Ainsi, en réponse à la requête de notre fil rouge, cela permettra ce créer un objet représentant l'alignement des deux séquences. Le schéma complet du module DsLSRBioObjects se trouve sur le CDROM (IMAGES\CLASS DIAGRAMS\DsLSRBioObjects.gif).

[15] Le type Any défini dans le langage IDL permet de représenter n'importe quel type de donnée là ou le type exact ne peut-être connu à priori

3.2 Module DsLSRAnalysis

Le module **DsLSRBioAnalysis** (figure 12) définit les interfaces nécessaires aux applications d'analyses de séquence en utilisant, et c'est très important, un modèle d'analyse générique.

Le module encapsule les éléments devant être utilisés pour ces analyses et fournit les moyens nécessaires à retrouver leurs inputs, outputs ou fonctionnalités d'un élément.

Un **AnalysisService** est une représentation logique d'un outil particulier d'analyse BSA disponible dans le système. Il peut par exemple provenir d'un logiciel ancien qui a été interfacé de manière à supporter la norme BSA. Les attributs d'un AnalysisService permettent l'identification d'un service par rapport à un autre grâce à l'information décrivant le type d'analyse, l'**AnalysisType,** ainsi que les inputs et outputs associés à ce service. Lorsqu'une requête lui est adressée, un AnalysisService crée et retourne une référence vers un objet **AnalysisInstance** qui implémente l'outil d'analyse BSA représenté. La description IDL du module DsLSRAnalysis hérite également des modules CORBA « CosLifeCycle » et « CosPropertyService » ainsi que du module «orb» et d'autres modules nécessaires à l'invocation asynchrone («TimeBase», «CosEventChannelAdmin»).

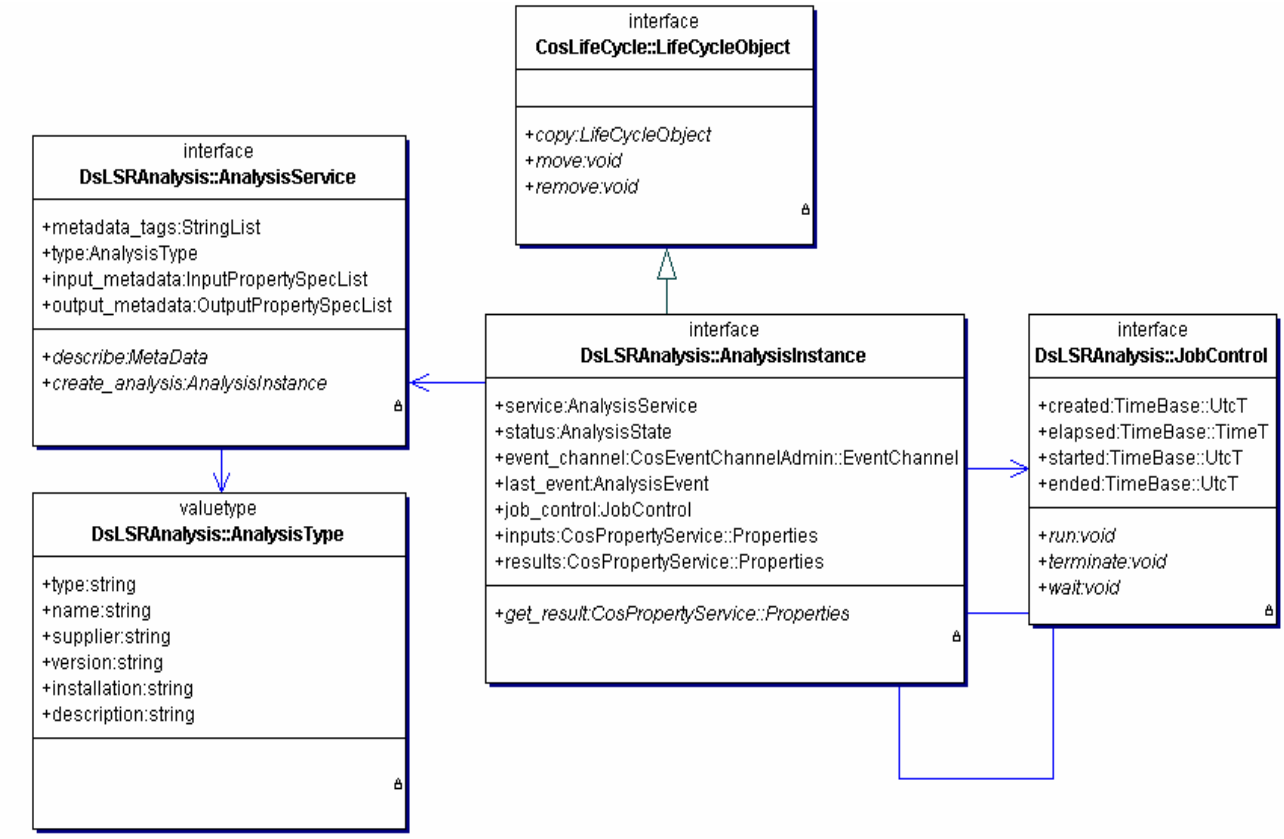

Figure 12 : Principales Interfaces du module DsLSRAnalysis

Le schéma complet du module sur le CDROM
(IMAGES\CLASS DIAGRAMS\DsLSRAnalysis.gif).

AnalysisService

Il s'agit de l'interface centrale du module d'analyse.
Elle contient l'information détaillant le service offert par une analyse en renfermant un AnalysisType et la description de ses inputs, outputs.

L'interface AnalysisService contient deux méthodes :

- *AnalysisInstance* ***create_analysis***
  (in CosPropertyService ::Properties input)

  Cette méthode contient les inputs nécessaire à une analyse sous la forme de propriétés CORBA «**properties** » et retourne un objet AnalysisInstance.
  Il est important de noter que c'est ici que se trouve la généricité d'une demande d'analyse dans la norme. En effet, cette méthode *create_analysis()* est commune à tous les AnalysisServices représentant tous les outils d'analyses disponibles. C'est en appelant la méthode générale *create_analysis()* sur un AnalysisService spécifique que l'on exécute une analyse particulière !

- *Metadata* ***describe*** *(in string tagname)*

  Selon le « tagname » utilisé à savoir input, output ou AnalysisType, cette méthode va permettre à un AnalysisService de renvoyer la description des inputs, outputs et AnalysisType sous forme d'une chaîne de caractères au format XML dont le type est appelé «**Metadata**».

AnalysisType

Un AnalysisType permet à un client de déterminer les types d'analyses BSA présentes dans un système. Il contient l'information telle que le type d'analyse, son nom, son fournisseur, sa version…. L'AnalysisType a été défini en tant que valuetype ; cela permet d'étendre le type en y ajoutant de nouveaux attributs.

InputPropertySpec

Un InputPropertySpec est utilisé afin de décrire les paramètres nécessaires aux inputs lors d'une analyse. Un input est défini notamment par son nom et son type. Il est à noter que le type associé est défini par le type CORBA «**typecode** »[16].

OutputPropertySpec

Il s'agit du même format de données que InputPropertySpec, mais appliqué aux outputs générés par une analyse.

[16] Le typecode permet de décrire tout les types standard IDL ainsi que les types définis par les utilisateurs. Le typecode sera utilisé pour obtenir la description d'un type quelconque d'un objet contenu dans un format « any ».

AnalysisInstance

L'AnalysisInstance est générée par un AnalysisService en réponse à un « *create_analysis()* ». Elle est responsable de l'invocation de l'outil d'analyse BSA qui effectue réellement l'analyse. Cette invocation est soit synchrone soit asynchrone. On ne peut exécuter qu'une seule fois un AnalysisInstance, si on désire répéter l'analyse avec d'autres inputs, il est nécessaire de créer une nouvelle AnalysisInstance via l'AnalysisService. Les attributs principaux d'un AnalysisInstance sont les inputs et résultats de l'analyse qui sont de la forme CORBA « properties ». Un AnalysisInstance contient également un objet appelé « **JobControl** ». Le JobControl, utilisé en mode synchrone ou asynchrone, est utilisé par le client afin de gérer le contrôle de l'exécution de l'analyse via des méthodes permettant de lancer la tâche et de se synchroniser sur la fin de celle-ci.

Le schéma ci-dessous (figure 13) résume une application élémentaire prévue par la norme (en mode synchrone). Dans ce schéma le client trouve les références d'un AnalysisService offrant le service désiré et il invoque la méthode *create_analysis()*. Cette méthode retourne une AnalysisInstance responsable de l'invocation auprès de l'outil d'analyse BSA.
L' AnalysisInstance crée à son tour un objet de contrôle d'exécution. Le client invoque l'objet de contrôle et se synchronise sur celui-ci. A la fin de l'exécution le client récupère les outputs de l'analyse en invoquant une méthode *get_results()* auprès de l'instance.

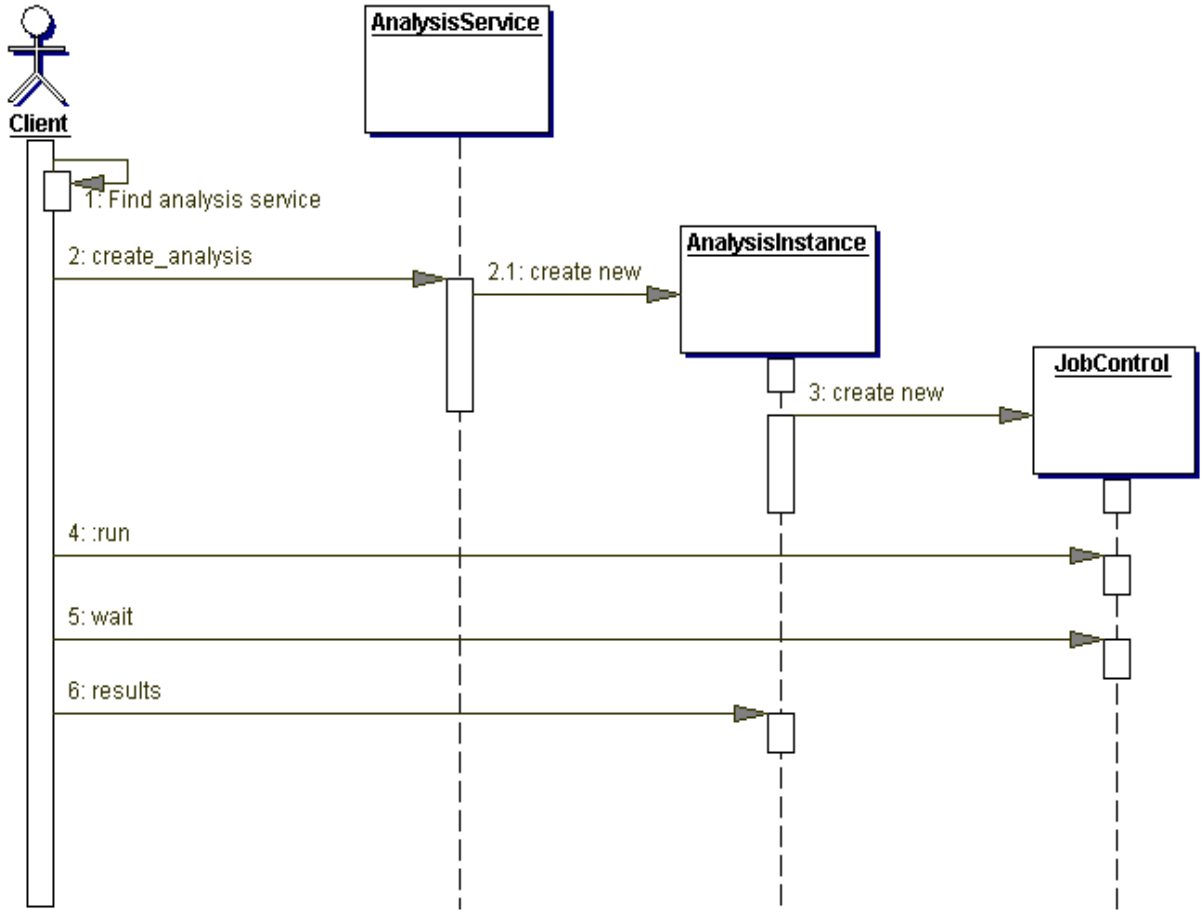

Figure 13 : Invocation synchrone

3.3 Responsabilités du client

Il est important de noter qu'il est de la responsabilité du client de :

- Déterminer l'outil d'analyse de séquence biomoléculaire BSA à utiliser
- Localiser un objet AnalysisService qui représente cet outil
- Obtenir et manipuler un objet AnalysisInstance qui implémente l'outil d'analyse BSA.
- Fournir l'information input nécessaire à l'AnalysisInstance
- Invoquer l'AnalysisInstance afin d'exécuter ses fonctions (en mode synchrone ou asynchrone)
- Soutirer les résultats produits par l'outil d'analyse
- Assurer la destruction des objets distants crées par l'analyse lorsque celle-ci est complètement terminée.

Chapitre 4

Analyse du projet

De manière très générale, le laboratoire de bioinformatique de l'ULB était désireux d'étudier la norme BioSequenceAnalysis de l'OMG et la faisabilité de sa mise en œuvre dans le cadre de la distribution des applications de bioinformatique et des bases de données génétiques. L'objectif final étant une utilisation plus aisée de la bioinformatique.

Après une brève étude de la situation actuelle et de la norme, nous nous sommes vite rendu compte qu'il était nécessaire de fractionner le travail vu son étendue. Il nous a semblé plus intéressant dans le cadre d'un mémoire de licence en informatique de nous focaliser sur le module DsLSRAnalysis de la norme plutôt que sur son module DsLSRBioObjects. En effet, étudier le module DsLSRBioObjects revenait à « formater » les données selon la norme BSA. Cette étude constitue nous croyons un travail pour des organismes tels que le l'EMBL qui désirent fournir des services en conformité avec la norme BSA et requiert plutôt des compétences en biologie. Le module DsLSRBioObjects pourra être étudié séparément après avoir défini une architecture décrivant le fonctionnement en système distribué. Par contre, le module DsLSRAnalysis, vu ses nombreux aspects non définis était plus propice à une étude approfondie. C'est donc sur lui que nous nous sommes concentrés. Nous avons aussi écarté du projet un essai d'encapsulation d'un legacy system pour le mettre en conformité avec la norme BSA. De même nous n'avons pas abordé le développement de l'interface Client de l'architecture, cette étude est en cours au BEN.

Ce travail a donc consisté au développement d'une architecture répondant à la norme BSA ainsi que la réalisation d'un prototype assurant la cohérence de son fonctionnement.

4.1. Analyse de la situation actuelle

Généralement, un biologiste accède via Internet et un navigateur à différents serveurs sur lesquels il peut utiliser diverses bases de données et logiciels d'analyses (figure 14).

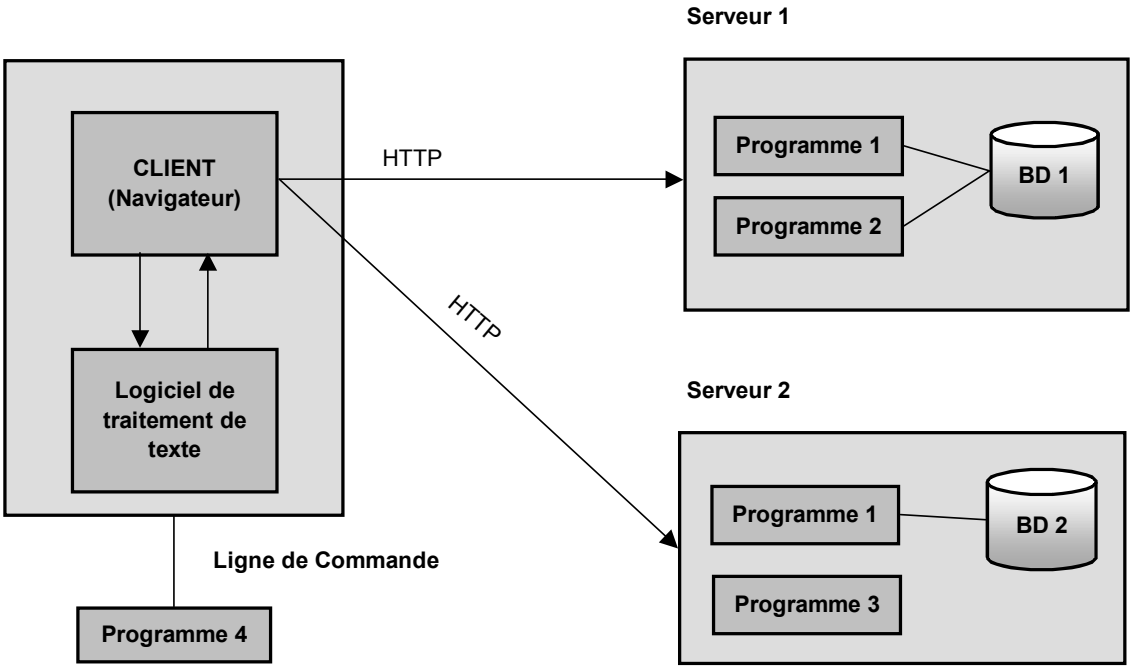

Figure 14 : Représentation de la situation actuelle

Ainsi, dans notre exemple du biologiste désirant comparer deux séquences, le biologiste doit d'abord se connecter au *serveur1* hébergeant la *base de données 1* sur laquelle il peut aller chercher les enregistrements correspondant aux séquences qu'il désire comparer. Une fois les enregistrements obtenus, il peut, par exemple copier la séquence dans un traitement de texte afin de la mettre au format désiré. Ensuite, il se connectera au *serveur2* afin d'utiliser le *programme3* à qui il fournira les séquences à comparer.

Cette situation peut être schématisée par les UseCases suivants :

UC1 : « Recherche de numéros d'accession sur base de mots clé »

Description : Le biologiste désire retrouver les numéros d'accession uniques de séquences voulues sur base de mots clé

Pré-condition : Le biologiste connaît un système permettant d'effectuer sa requête

Biologiste	Système
Le biologiste sélectionne un système	
	Le système renvoie les BD et logiciels d'interrogation accessibles
Le biologiste sélectionne la base de données et le logiciel de recherche qu'il désire utiliser et entre les mots clés	
	Le système renvoie les numéros d'accession uniques correspondants

(fin UC1)

Post-condition : Le système a renvoyé les numéros d'accession uniques correspondant à la requête

UC2 : « Lire un enregistrement » **uses UC1**

Description : Le biologiste va chercher l'enregistrement correspondant à un n° d'accession donné et en isole la séquence

Pré-condition : Le numéro d'accession de l'enregistrement est connu

Biologiste	Système
Le biologiste entre un numéro d'accession	
	Le système renvoie l'enregistrement correspondant.
Le biologiste isole la séquence dans un format donné.	

(Fin UC2)

UC3 : « Alignement » **uses UC2**

Description : Le biologiste désire aligner deux séquences en fournissant au logiciel ad hoc les séquences à aligner

Pré-conditions : Le biologiste possède les séquences au format nécessaire au logiciel qu'il désire utiliser.
Le biologiste connaît un système permettant d'effectuer sa requête

Biologiste	Système
Le biologiste sélectionne un système	
	Le système renvoie les logiciels disponibles
Le biologiste sélectionne le logiciel nécessaire et lui fournit les deux séquences à aligner	
	Le système renvoie les deux séquences alignées

(Fin UC3)

Post-condition : Les deux séquences entrées sont correctement alignées

Cet exemple illustre quelques uns des problèmes de la bioinformatique actuelle

- Les environnements sont très hétérogènes : hardware, systèmes d'exploitation, programmes, interfaces, langages, formats des données,…
- Il existe une inconsistance dans la modélisation et la sémantique de l'information.
- Il incombe au biologiste de localiser les serveurs où sont disponibles les différentes bases de données et logiciels.
- Les analyses en bioinformatique étant souvent constituées d'une succession d'étapes réalisées avec des logiciels différents, il est usuel de « reformater » les données d'une application à l'autre. Ceci rend difficile le développement de scripts enchaînant les différentes étapes.

4.2. Vers une utilisation simplifiée de la bioinformatique

L'utilisation de CORBA et de la norme BSA de l'OMG permettrait de résoudre beaucoup de ces problèmes. En effet, CORBA assure la transparence de la localisation et des accès, masque l'hétérogénéité des softwares [Coupaye 99] tandis que la norme BSA propose un format standard de données et un modèle simplifié d'analyse.

L'exemple présenté plus haut pourrait alors se modéliser par ce UseCase simplifié

UC Simplifié : « Analyse »

Description : le biologiste désire aligner deux séquences obtenues par une recherche sur base de mots clés.

Pré-condition : /

Biologiste	Système
Le biologiste se connecte au système	
Le biologiste sélectionne l'outil de recherche et introduit les mots clé.	
	Le système localise la base de données.
	Le système effectue la requête et renvoie les enregistrements trouvés.
Le biologiste sélectionne les enregistrements dont il désire comparer les séquences et soumet la demande de comparaison au système.	
	Le système localise un serveur capable d'effectuer la comparaison demandée
	Le système effectue la requête et renvoie le résultat de la comparaison au biologiste.

(fin UC)

Post-condition : Les deux séquences entrées sont correctement alignées

Remarque

Le biologiste n'a plus eu à se soucier ni des problèmes de localisation des données ni des problèmes d'interface et de formats de données.

4.3 Conception d'une architecture générale

Nous allons dans un premier temps présenter l'architecture globale (figure 15) qui s'est dégagée après les différentes réunions de travail organisées au laboratoire de Bioinformatique

de l'ULB et de la lecture de la littérature ainsi que de la norme BSA. Au fur et à mesure des questions qui se sont posées, nous allons affiner cette architecture.

Selon la norme BSA, les fonctionnalités des legacy systems sont encapsulées dans les AnalysisServices. Nous utiliserons le terme **wrapper** pour parler de l'élément comprenant l'AnalysisService mais également les fonctionnalités ajoutées pour le bon fonctionnement de notre architecture. Un même wrapper peut être localisé sur plusieurs serveurs (pouvant héberger différents wrappers).

Afin de couvrir les aspects non définis dans la norme BSA (localisation d'un AnalysisService particulier, invocation d'une AnalysisInstance implémentant le service, fourniture des inputs nécessaires et récupération des résultats), nous avons rajouté un élément au système: le **médiateur**. De plus, le médiateur pourra à l'avenir, à l'instar des couches réseau, avoir un fonctionnement en couche. Il se verrait ainsi confier la gestion des accès, de la sécurité et de la performance. Il pourrait également proposer de nouveaux services résultants de l'intégration de plusieurs services existants (par exemple un script faisant se succéder l'analyse A par l'analyse B qui prendrait en inputs les outputs de l'analyse A)
Ce médiateur sera le point de contact unique des wrappers lorsque ceux-ci voudront s'intégrer dans le système. De même, le médiateur sera le point de contact unique des clients désirant adresser une requête au système. Le médiateur a donc un rôle primordial dans notre architecture. Quand un client soumet une requête au médiateur, celui-ci connaissant tous les AnalysisServices disponibles (et leur localisation), il peut la transmettre à l'AnalysisService ad hoc.

Les **clients** adressent leurs requêtes au médiateur conformément à la norme BSA car le médiateur est « vu » comme un ensemble d'AnalysisServices (dans un premier temps)

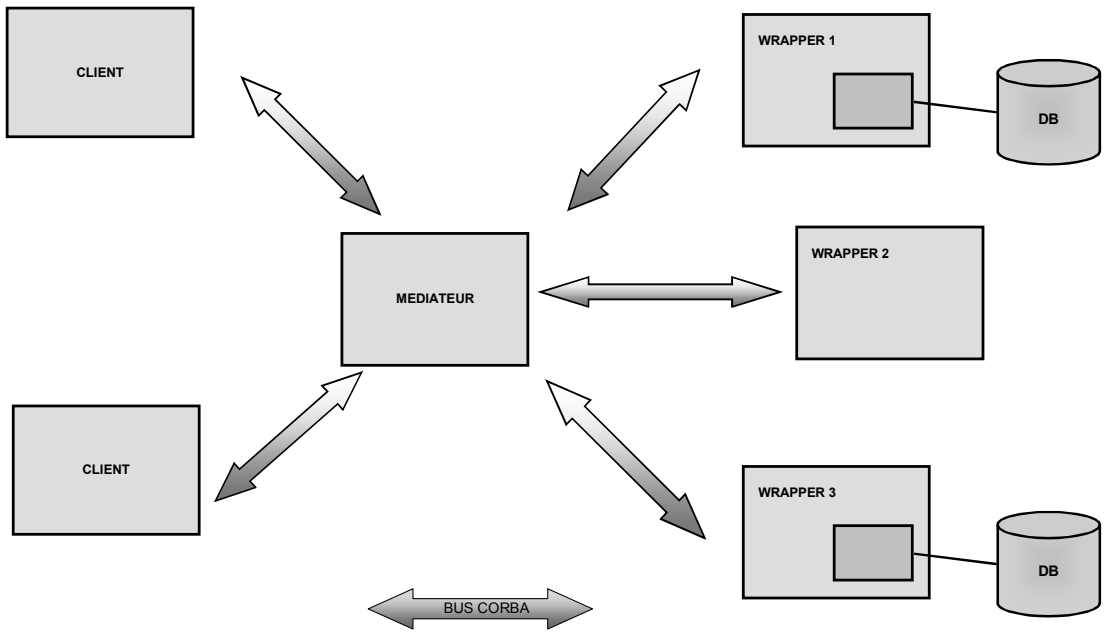

Figure 15 : Architecture générale à un médiateur

Cette ébauche d'architecture soulève plusieurs questions :

1. Comment les wrappers se déclarent-il au médiateur ?
2. Comment les wrappers et les client connaissent-ils l'IOR du médiateur ?
3. Comment le médiateur gère-t-il les IOR des wrappers connus ?
4. Comment les clients connaissent-ils l'ensembles des AnalysisServices disponibles ?
5. Comment comparer deux AnalysisServices pour déterminer s'ils fournissent le même service ?
6. Comment les clients adressent-ils une requête au médiateur ?
7. Comment le médiateur répercute-t-il une requête au wrapper ad hoc?

Nous allons répondre à ces questions une à une en affinant au fur et à mesure notre architecture.

Question 1. Comment les wrappers se déclarent-il au médiateur ?

Afin de permettre aux wrappers de pouvoir se déclarer au médiateur et décrire le service qu'ils proposent, nous avons appliqué le mécanisme de base décrit dans la norme, mécanisme qui a été dû être complété pour tenir compte de notre architecture et des aspects non définis dans la norme BSA (figure 16).
La norme propose qu'une fois un AnalysisService localisé, on peut lui appliquer une méthode `describe()` qui renverra les caractéristiques du type d'analyse réalisée. Mais pour pouvoir faire appel au `describe()` d'un AnalysisService, il faut connaître sa localisation (son IOR) ! Nous avons donc rajouté une méthode `hello()` au médiateur. Cette méthode est appelée à l'initialisation du wrapper qui signale ainsi son apparition dans le système. Ainsi le médiateur peut, connaissant l'IOR de l'AnalysisService reçu par la méthode `hello()`, appeler la méthode `describe()` sur l'AnalysisService de ce wrapper. Pour pallier à une défaillance du médiateur ou au redémarrage du serveur l'hébergeant, ce `hello()` pourrait être répété à intervalle régulier.

L'AnalysisService est alors « connu » et localisé par le système, en l'occurrence le médiateur à ce stade.

Dans le but de concevoir une architecture modulaire et réutilisable nous avons ajouté un composant à l'AnalysisService sur l'hôte du wrapper, le **Front-End** et un nouveau composant au médiateur, l'**Office**

Le Front-End :
Il s'agit d'un composant du wrapper qui a pour but de prendre en charge le dialogue avec le système -dans ce cas le médiateur- afin notamment de lui adresser un « `hello()` » et entamer le protocole de communication.

L'Office :
C'est par son Office qu'un médiateur va correspondre avec les Front-ends des wrappers.

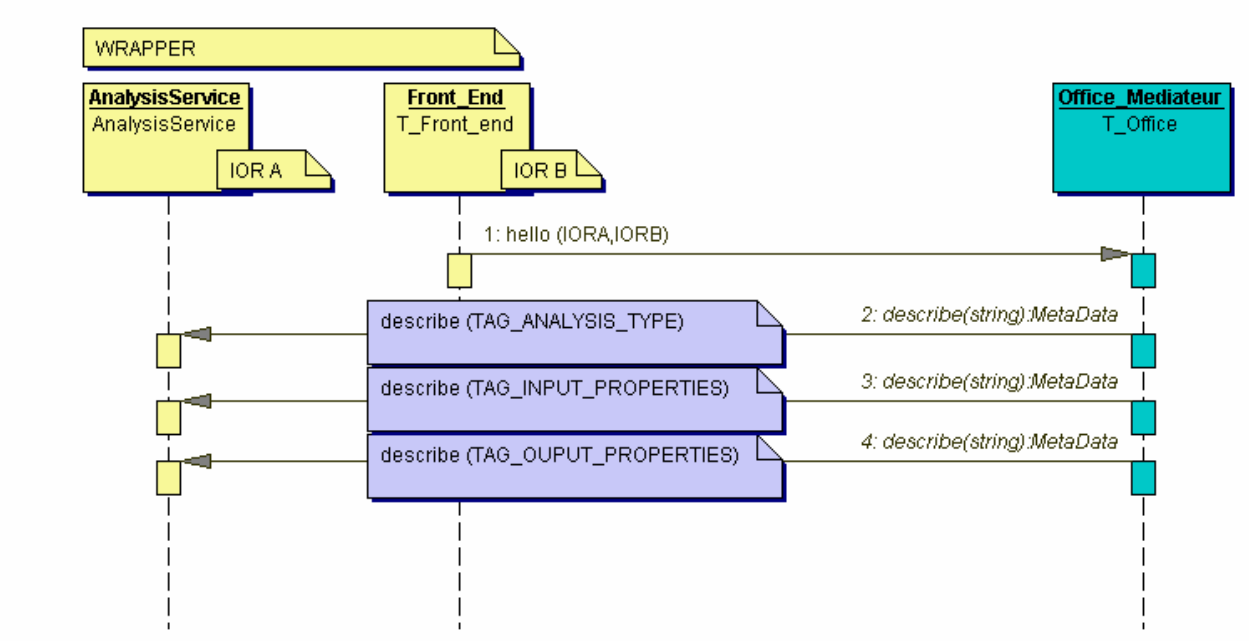

Figure 16 : Sequence diagram illustrant la déclaration d'un wrapper au médiateur

NB : Les éléments nouveaux non décrits dans la norme BSA et introduits à ce stade sont :
Le Front_End du wrapper
L'Office du médiateur
La méthode `hello()`

Question 2. Comment les wrappers et les clients connaissent-ils l'IOR du médiateur ?

L'IOR du médiateur serait connu des wrappers et des clients via sa publication sur une page web dont l'URL serait connue de tous.

Question 3. Comment le médiateur gère-t-il les IOR des wrappers connus ?

Nous avons vu plus haut que ce sont les wrappers qui viennent s'identifier auprès du médiateur. En pratique les IOR des wrappers peuvent varier: redémarrage du serveur, changement de machine, mais il peut aussi s'ajouter de nouveaux services, etc.
Ces IOR et la description des services (AnalysisTypes) sont repris dans un **IOR Repository** intégré au médiateur. Cet IOR Repository permet donc, connaissant un AnalysisService de retrouver son IOR. Le médiateur a pour tâche de maintenir l'IOR Repository et peut de la sorte connaître en temps réel l'ensemble des AS disponibles dans le système.

Question 4. Comment les clients connaissent-ils l'ensemble des AnalysisServices disponibles ?

Les AnalysisServices disponibles, actifs sont ceux qui se sont déclarés au médiateur et qui ont alors été ajoutés à l'IOR Repository. Nous avons imaginé un système (figure 17) où, lorsqu'un client s'initialise, il demande au médiateur via la méthode

`get_Active_Services()` de lui renvoyer les AnalysisServices connus dans l'IOR Repository.

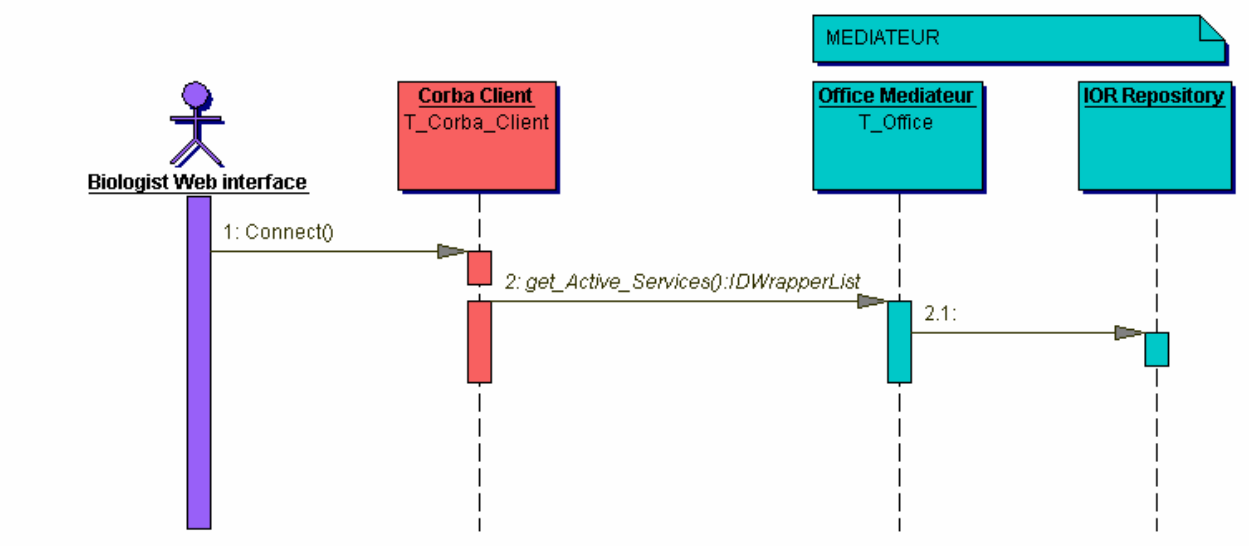

Figure 17 : Sequence diagram illustrant la recherche des services actifs

NB : Les éléments nouveaux non décrits dans la norme BSA et introduits à ce stade sont :
- L'IOR Repository
- La méthode `Connect()`
- La méthode `get_Active_Services()`

Notons que le **client corba** peut recevoir les requêtes d'un **client web** via le protocole HTTP (ce qui se fera le plus souvent en pratique).

Question 5. Comment comparer deux AnalysisServices pour déterminer s'ils fournissent le même service ?

Pour rappel, la norme BSA spécifie plusieurs attributs à l'AnalysisType permettant de caractériser un AnalysisService :
- Type : permet de classer les analyses en fonction d'une classification préétablie
- Name : utilisé pour nommer l'analyse dans le système
- Supplier : identifie le responsable de l'implémentation de l'AnalysisService
- Version : numéro de version de l'implémentation
- Installation : permet de spécifier la localisation d'une installation d'un AnalysisService
- Description : Décrit le service rendu

Cette liste peut être étendue en fonction des besoins.

Ceci nous amène à constater que pour pouvoir comparer des AnalysisServices, il faut uniformiser, standardiser la façon de décrire ces AnalysisServices via leur AnalysisType. En effet, le même AnalysisService doit toujours être décrit par le même AnalysisType. Il nous est alors paru opportun de créer un repository universel que nous avons appelé **Master Definition Repository (MDR).**

A ce stade se pose la question de savoir comment considérer deux services à priori identiques. Par exemple une recherche sur la base de données GenBank sur base d'un accession number avec un wrapper encapsulant cette fonction du logiciel SRS, mais situés sur des serveurs différents sont-ils des AnalysisServices différents ou identiques ? En d'autres termes, quels sont les attributs de l'AnalysisType à prendre en compte pour déterminer l' « identifiant ».
La question de la composition de l'identifiant d'un AnalysisService reste posée, mais nous pensons qu'il est préférable que cette identification soit la plus précise possible, ce qui permettra par après de faire le choix d'un AnalysisService en fonction de différents critères (performance, etc.) De même, à ce stade de l'étude nous avons, en accord avec le BEN, formulé une hypothèse importante : *un wrapper est déterministe* c'est à dire qu'une même requête faite plusieurs fois fournira toujours la même réponse.

Fonctionnement du MDR

Vu son aspect universel, le fonctionnement précis du MDR (figure 18) doit faire l'objet d'une étude spécifique (par exemple à l'aide du BEN). Les éléments principaux à prendre en considération seront bien sûr la structure des données servant à identifier les services offerts par les wrappers (description des AnalysisTypes, des inputs et outputs associés , ainsi que les directives nécessaires aux utilisateurs de services). L'accès aux données devra lui aussi faire l'objet d'une étude particulière (probablement via le web). Quels sont les utilisateurs qui doivent être prévenus en cas d'ajout, de modification ou de suppression de services (mécanisme « d'abonnement aux événements » à implémenter) ?

Le MDR doit également être unique, cela engendre un problème d'administration : qui va gérer ce repository, en garantir l'intégrité et la persistance des données, qui va donner les droits d'accès en lecture ou en écriture ? Enfin, citons les habituels problèmes de sécurité liés aux attaques extérieures ...

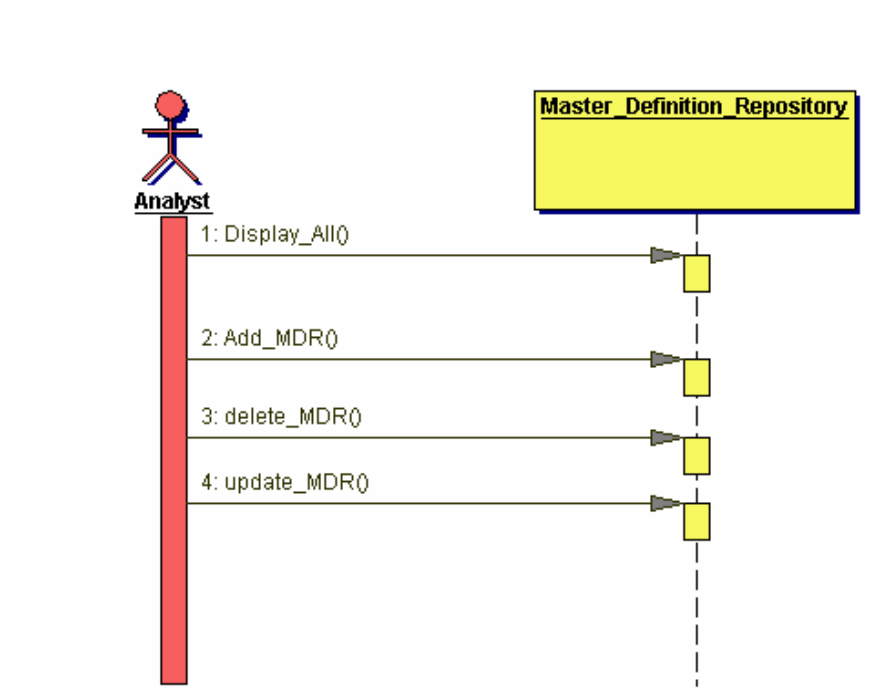

Figure 18 : Sequence diagram schématisant le fonctionnement du MDR

NB : Aucun de ces éléments ni méthodes ne sont décrits dans la norme BSA

Question 6. Comment les clients adressent-ils une requête au médiateur ?

Par rapport à la norme BSA nous avons ajouté un élément fondamental : le médiateur. C'est à lui que les requêtes doivent à présent être adressées comme si le client s'adressait directement à un AnalysisService (figure 19). Ainsi, notre médiateur peut être utilisé par tout client, développé ou non par nous, vu sa conformité avec la norme BSA.

La norme BSA stipule qu'un client doit fournir une requête à un AnalysisService à l'aide de la méthode `create_analysis()`. Ce `create_analysis()` doit être directement adressé à l'AnalysisService désiré, après l'avoir choisi et localisé. Un client ne peut donc effectuer son `create_analysis()` sur l'office du médiateur.
Pour résoudre ce problème, lorsque chaque wrapper se déclare au médiateur, nous y créons une « image » de son AnalysisService que nous appelons **proxy.** (Il y aura donc autant de proxys que d'AnalysisServices connus par le médiateur.) L'IOR du proxy est gardé dans l'IOR Repository en association avec l'AnalysisService correspondant et l'IOR du wrapper. Ce proxy est donc l'entité sur laquelle les clients pourront effectuer les `create_analysis()`.
Quand un client désire maintenant utiliser un service particulier, il demande l'IOR de ce service au médiateur qui renverra l'IOR du proxy correspondant à ce service.

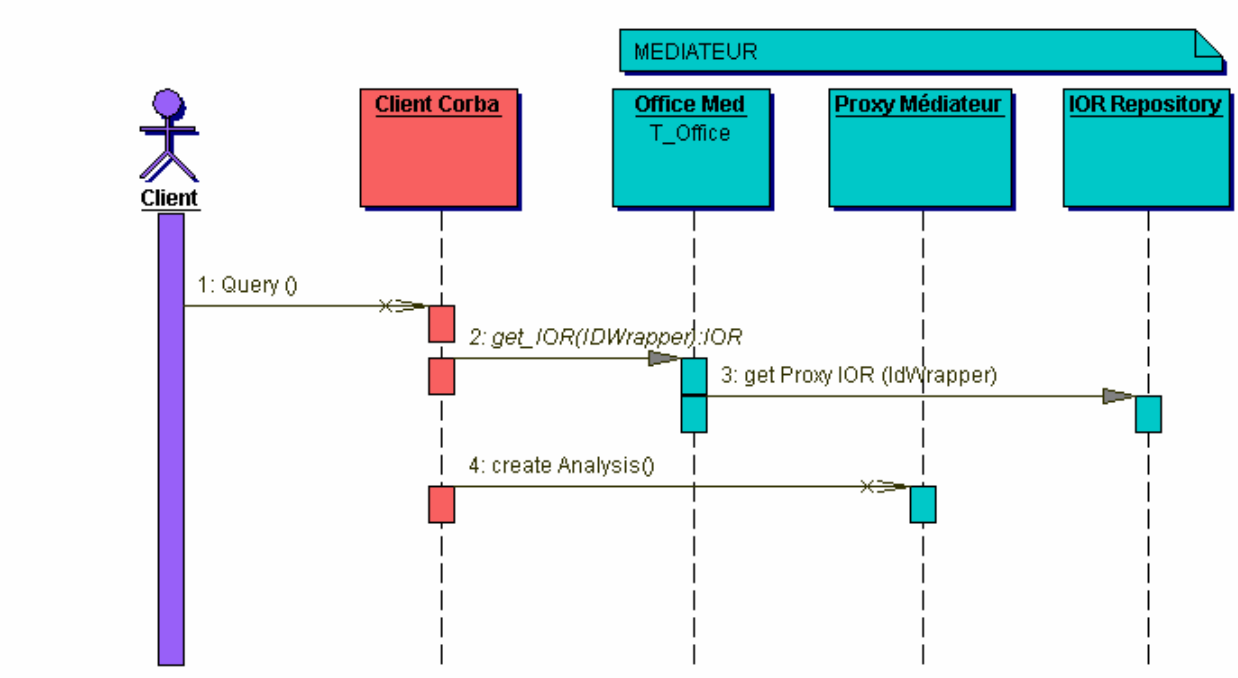

Figure 19 : Sequence diagram illustrant une demande d'analyse au médiateur

NB : Les éléments nouveaux non décrits dans la norme BSA et introduits à ce stade sont :
- Les Proxys
- La méthode Query()
- La méthode get_IOR()
- La méthode get Proxy IOR()

A ce stade, on voit donc que le client peut s'adresser au seul médiateur afin d'effectuer toutes les analyses disponibles sur le système. Nous avons donc rendu la localisation des wrappers transparente pour les clients.

Question 7. Comment le médiateur répercute-t-il une requête au wrapper ad hoc ?

En réponse à un `create_analysis()`, l'AnalysisInstance générée par et sur le médiateur se chargera de propager ce `create_analysis()` auprès du wrapper ad hoc (figure 20). Pour ce faire, l'AnalysisInstance générée par le proxy consulte l'IOR Repository et connaissant son AnalysisType va chercher l'IOR de l'analysisService. Grâce à cet IOR, l'AnalysisInstance du proxy peut effectuer le `create_analysis()` sur l'AnalysisService du wrapper ad hoc.

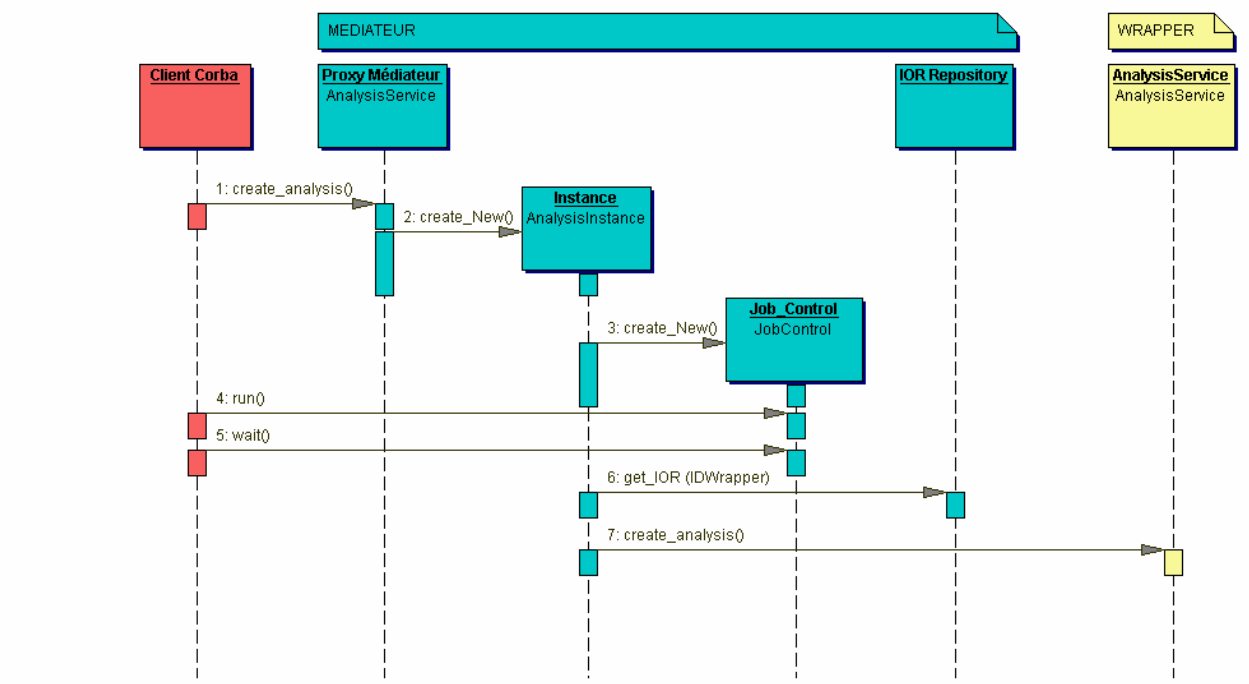

Figure 20 : Sequence diagram illustrant la transmission d'une requête à travers le système

Remarquons que l'implémentation des AnalysisInstances générées par les proxys est identique quel que soit l'AnalysisService représenté. En effet, l'AnalysisInstance doit simplement consulter l'IOR Repository afin de trouver l'IOR de l'AnalysisService auquel il correspond et répercuter la requête auprès de cet AnalysisService.

4.4 En route vers une architecture définitive

Bien que le projet proposé à ce stade nous semblait abouti, certaines exigences nous ont amené à compléter l'architecture. Ainsi, comment assurer la diffusion restreinte de certains AnalysisServices ? En d'autres termes, comment une société privée peut-elle utiliser notre architecture afin de profiter des services proposés universellement, mais rendre l'accès à certaines données internes confidentielles voire payantes ?

De fait, ne peut-on pas envisager que notre architecture puisse fonctionner simultanément avec plusieurs médiateurs (un par « site »), ces médiateurs gérant également les accès aux services d'un site donné? Notre projet franchit ainsi une nouvelle étape, le fonctionnement en fédération.

4.5 Conception d'un système fédéré dynamique

Schématiquement, afin de répondre à l'exigence de partage restreint des services, chaque médiateur, lorsqu'il incorpore la fédération, est libre de divulguer ou non ses services (figure 21) .

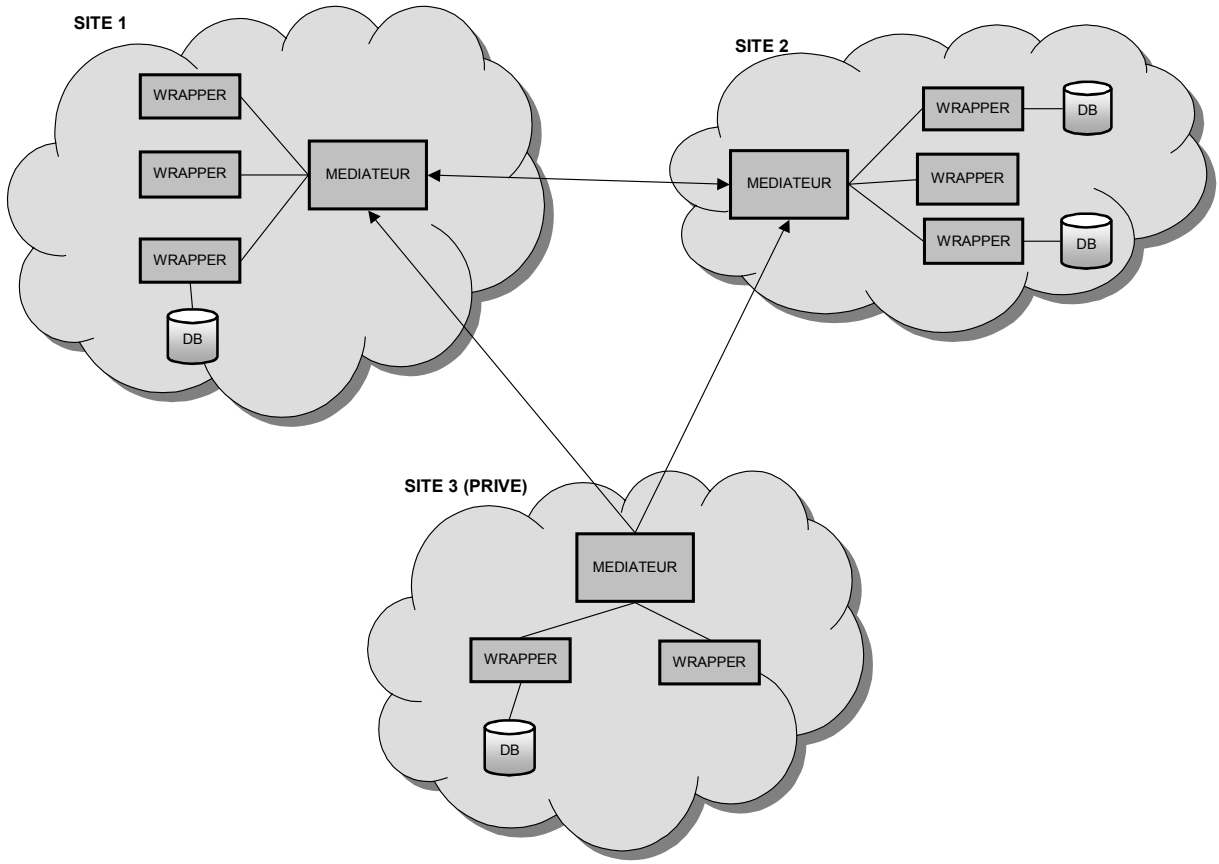

Figure 21 : Architecture générale à plusieurs médiateurs

Dès ce moment, une nouvelle difficulté se pose : auparavant un wrapper pouvait se déclarer auprès du seul médiateur qui avait en charge l'inscription du service rendu dans l'IOR Repository. Comment un wrapper maintenant va-t-il se déclarer auprès de la fédération ? Il ne parait pas réaliste et performant qu'un wrapper effectue cette fois le `hello()` auprès de tous

les médiateurs existants (comment connaître leur adresse – aspect dynamique). Nous avons levé ce problème en imaginant un réseau de médiateurs totalement dynamiques qui s'échangeraient les informations connues entre eux en fonction de leurs « affinités ». Ainsi, un wrapper n'a plus qu'à se déclarer auprès du médiateur de son organisation qui répercutera cette information ou non aux autres médiateurs. En d'autres termes, nous pensons que chaque médiateur doit avoir un « carnet d'adresses » précisant qui ils doivent prévenir lorsqu'ils sont avertis de l'arrivée d'un nouveau wrapper ou de la modification d'un IOR. Vu que ce genre de relations s'effectuera entre les Offices des médiateurs, nous avons appelé ce carnet d'adresses le « **Friend Office IOR Repository** »

Nous pouvons a présent (figure 22) compléter le schéma relatif à l'initialisation d'un wrapper (dans un but de clarté nous n'avons repris que la partie intervenant lorsqu'un wrapper a déjà fait son `hello()` à un médiateur, de même les arguments des fonctions ne sont pas détaillés – le lecteur consultera le schéma complet repris en annexe).

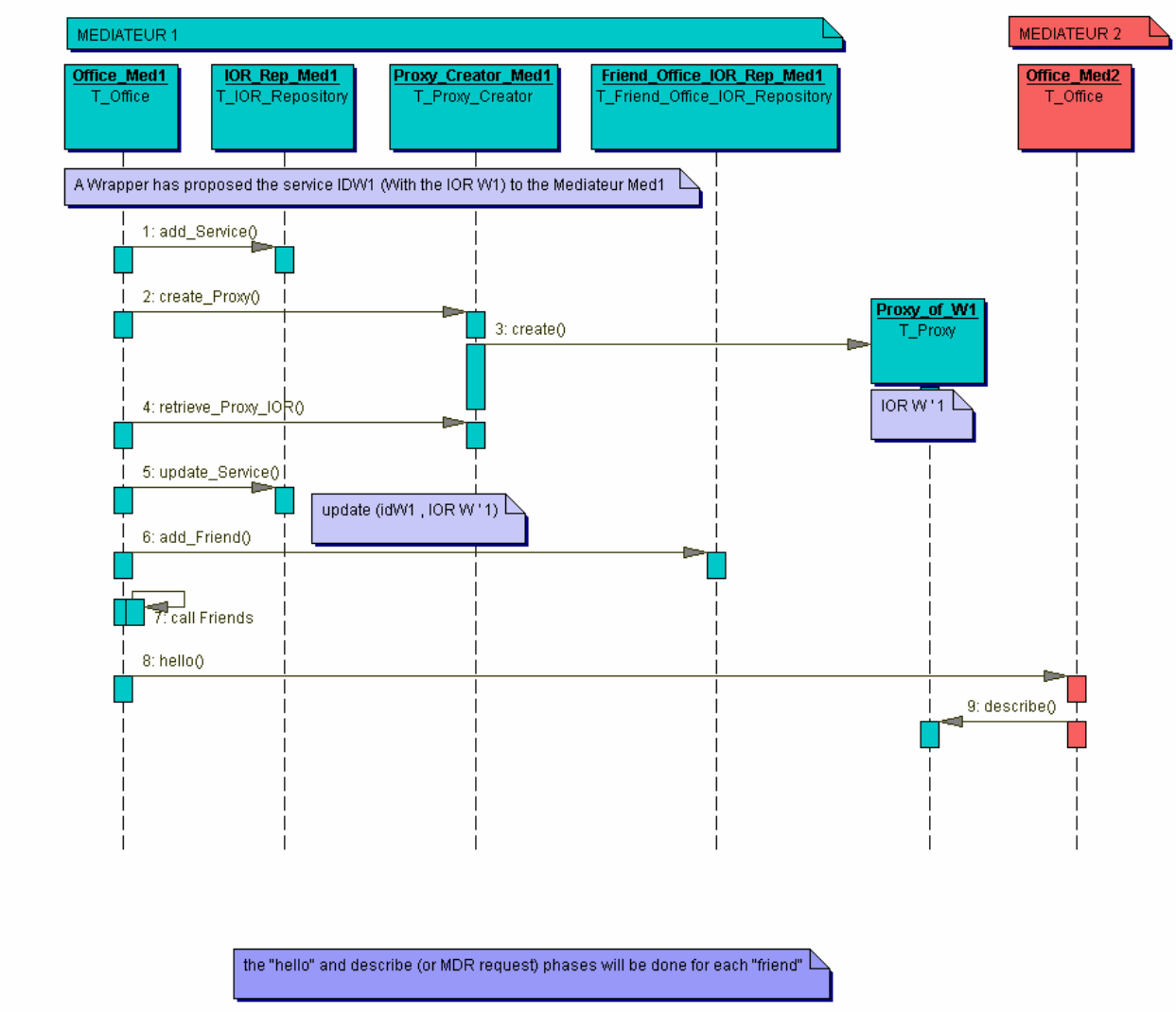

Figure 22 : Sequence diagram illustrant la communication des services entre médiateurs

Nous remarquons qu'après avoir été contacté par un wrapper pour un service idw1, le médiateur commence par ajouter ce service dans son IOR Repository. Il s'adresse alors à une entité de type « Proxy_Creator » servant à créer le service proxy correspondant au service offert par le wrapper. L'IOR du proxy (w'1) lui est retourné par la méthode « `retrieve_Proxy_IOR()` , et sera ajoutée dans son IOR Repository par la méthode `update_Service(idw1, IOR w'1).`
Dans ce schéma nous supposons qu'un médiateur « ami » a été inscrit dans le Friend IOR Repository, l'ajout se faisant par la méthode `add_Friend().`

Dans le cas d'un nouveau service offert par un wrapper ou en cas de rédémarrage du serveur hébergeant un wrapper, le médiateur consulte son Friend Office IOR Repository pour savoir quels sont les médiateurs amis à qui il doit répercuter cette information. L'office du médiateur appelle la méthode `hello()` de l'office du ou des médiateurs amis en lui transmettant l'IOR du proxy correspondant. Ainsi, en réponse à ce `hello()`, et conformément à la norme BSA, un `describe()` pourra être appelé sur le proxy pour connaître le type de service, les inputs et outputs.

Le médiateur ami peut à son tour contacter ses propres amis via ce même processus récursif. Comme cela, un médiateur connaîtra toujours tous les services offerts par la fédération plus ceux éventuellement non distribués, confidentiels de son site.

4.6 Transmission d'une requête à travers le système

Nous venons de décrire tous les éléments constitutifs de notre architecture. Rappelons que les proxys, vus de l'extérieur, semblent avoir le même comportement qu'un AnalysisService (nous verrons plus loin dans le chapitre concernant le prototype que l'interface proxy hérite de l'interface AnalysisService).
Deux cas de figure peuvent se présenter :

- un médiateur peut avoir a été directement contacté par un wrapper, auquel cas il pourra lui soumettre directement la requête le concernant et renvoyer directement les résultats à son client.
- un médiateur peut avoir été averti du service offert via un médiateur ami. Dans ce cas, il répercutera la demande au médiateur ami qui lui même la répercutera directement à un wrapper ou encore vers un nouveau médiateur ami etc...

Le fonctionnement correspond bien à un fonctionnement en fédération dynamique .Pour autant que chaque médiateur se conforme à notre architecture, il n'y a aucune contrainte dans l'ajout de nouveaux médiateurs ou dans la sophistication de ceux-ci en vue de répondre à des services de plus en plus complexes (exemple : la soumission de requêtes composites ou le résultat dépend du résultat d'autres services).

Les diagrammes ci-dessous illustrent le fonctionnement du système lors de la déclaration d'un wrapper et d'une demande d'analyse par un client.

Etape 1 : *hello()* d'un wrapper (figure 23)

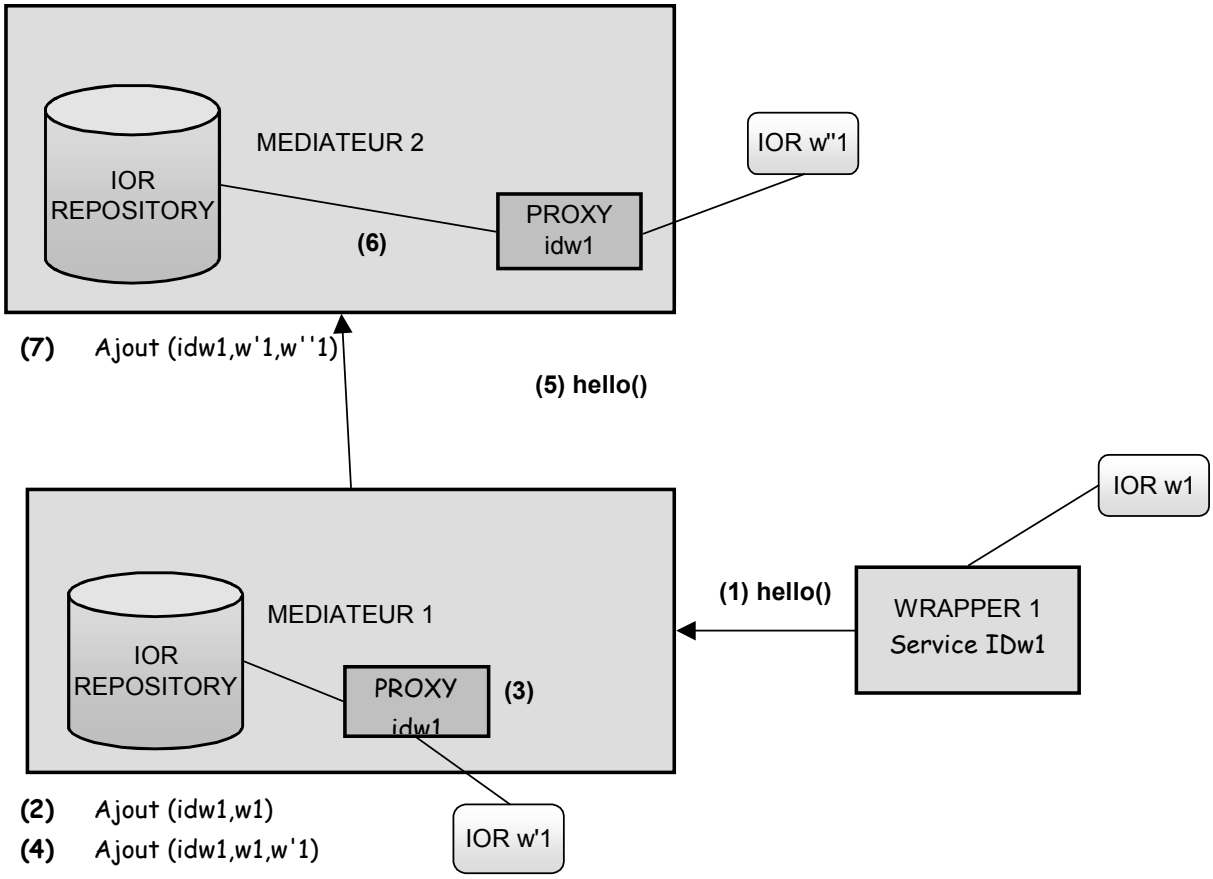

Figure 23 : Mises à jour de l'IOR Repository des médiateurs

1. Un Wrapper identifié idw1 et dont l'IOR est w1 fait un *hello()* auprès d'un médiateur1.
2. L'IOR Repository du médiateur1 contient maintenant le lien (idw1,w1)
3. Le médiateur1 crée un proxy dont l'IOR est w'1
4. Le médiateur1 complète son IOR Repository avec le lien (idw1,w1,w'1) – soit identification du service, adresse à laquelle il peut envoyer le *create_analysis (« forward » IOR, w1)* et adresse proxy (w'1).
5. Le médiateur1 averti un médiateur ami ,le médiateur2 en l'informant qu'il peut le mettre en contact avec le service idw1 à l'adresse de son proxy, soit w'1
6. Le médiateur2 crée un proxy de idw1 dont l'IOR est w''1
7. Son repository sera completé par le lien (idw1, w'1,w''1), c'est à dire identification du service, « forward IOR, w'1 » soit le proxy au sein du médiateur1, et IOR de son propre proxy (w''1).

Ce processus est répété de médiateur en médiateur.

Etape 2 : *create_analysis()* d'un client au médiateur2 (figure 24)

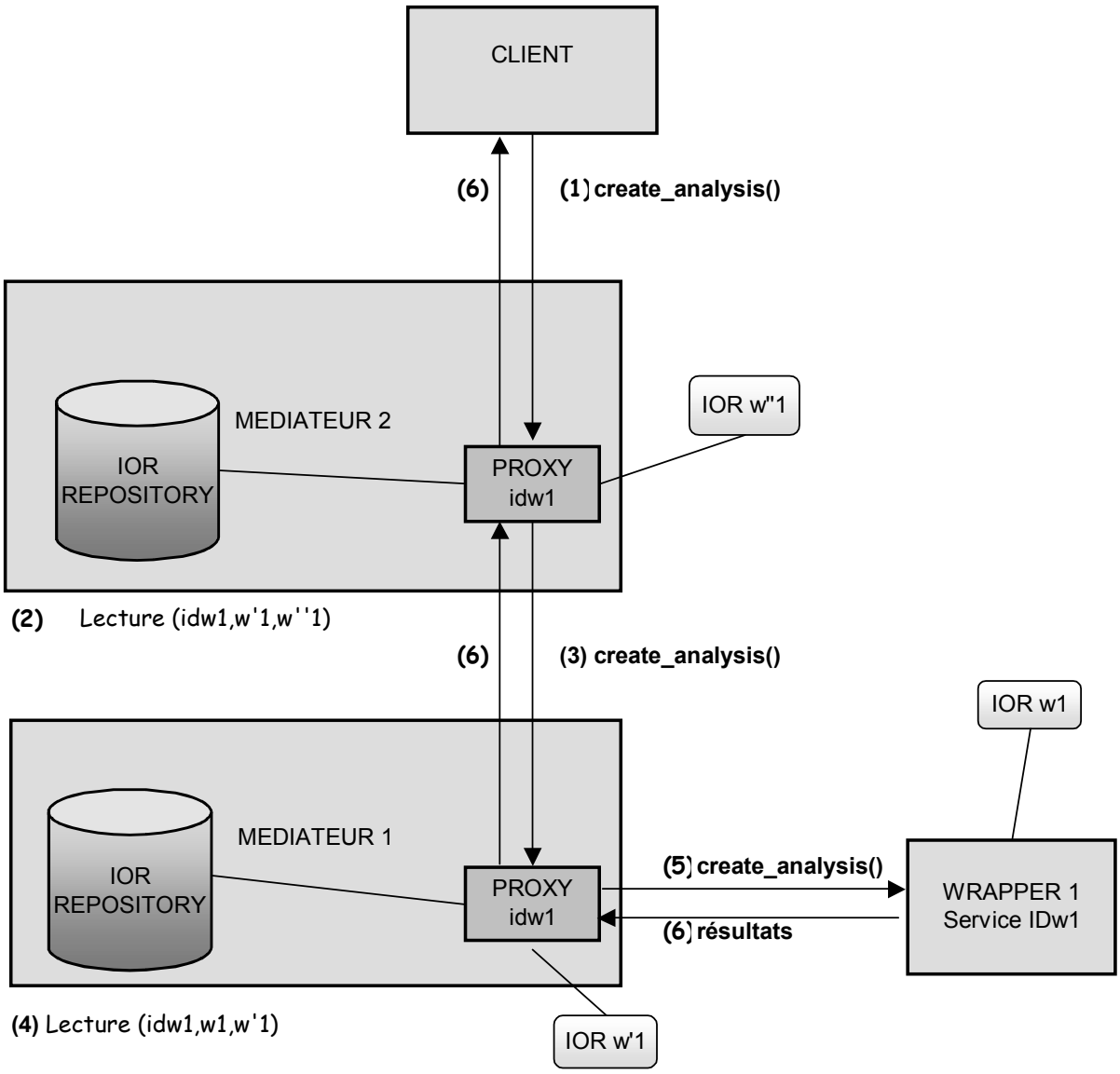

Figure 24 : Répercussion du *create_analysis()*

1. Après avoir consulté le médiateur 2 comme vu précédemment, le client désire adresser un *create_analysis()* concernant un service idw1 au médiateur2.
   Il contacte le médiateur2 qui lui renvoit l'IOR de son proxy (w''1)
   Le client effectue alors le *create_analysis()* auprès du proxy w''1.
2. L'AnalysisInstance créée par le proxy consulte l' IOR Repository du médiateur2 et constate qu'il peut répondre au service idw1 à l'aide du « forward » IOR w'1.
3. Il répercute alors le *create_analysis()* à l'IOR w'1 qui arrive au proxy du médiateur1

4. Le proxy du médiateur1 reçoit un `create_analysis()` concernant un service idw1. L'AnalysisInstance créée par le proxy du médiateur1 consulte l'IOR Repository et trouve le « forward »IOR w1.
5. Il répercute le `create_analysis()` à l'IOR w1 qui arrive au wrapper qui encapsule le service qui va réellement effectuer l'analyse.
6. Les résultats sont propagés du wrapper vers le médiateur1, puis vers le médiateur2 qui les renvoie au client.

Ce processus s'appliquera de la même manière quel que soit le nombre de médiateurs.

On observe au travers ces exemples qu'un wrapper peut aisément se faire connaître à toute la fédération. De même, un client peut adresser une requête à un médiateur qui se chargera de répercuter cette requête à travers la fédération jusqu'au destinataire final capable d'exécuter cette requête.

Le sequence diagram complet modélisant la requête d'un client à un médiateur se trouve en annexe.

Chapitre 5

Réalisation du prototype

Dans ce chapitre, nous allons développer la méthodologie employée afin d'implémenter un prototype de notre architecture. Pour ce faire, nous avons utilisé le langage (JDK 1.3.1_01) et l'ORB ORBacus 4.1.0[17] de la firme IONA Technologies.

5.1 Objectifs du Prototype

Il nous a semblé intéressant de réaliser un prototype sur base de l'architecture que nous venons de proposer. Ce prototype aura pour objectif de vérifier la cohérence de cette architecture et pourra servir de base à un développement futur (cf le développement « en couches » proposé).

Rappelons que nous ne nous sommes pas du tout intéressés au module DsLSRBioObjects (modélisant l'aspect « données ») mais bien au module DsLSRAnalysis (modélisant les mécanismes d'analyses). De même, le Master Definition Repository n'a pas été implémenté et le wrapper développé va simuler la tâche demandée et renvoyer un résultat sous forme de properties pré- encodées. Afin de simplifier le prototype nous avons envisagé le fonctionnement avec un seul médiateur et un seul wrapper, mais l'étendue à plusieurs médiateurs et wrappers est – selon notre architecture – automatiquement dérivée.

Le scénario global implémenté est le suivant :

Soit un médiateur instancié, un wrapper vient s'y déclarer via la méthode `Hello()`. Ensuite, un client soumet une requête à l'aide de la méthode `create_analysis()` sur le proxy créé au sein du médiateur. Cette requête sera répercutée au wrapper qui renverra un message au client via le médiateur. (Dans un fonctionnement pratique le wrapper devra interfacer un legacy system afin d'exécuter réellement une requête.)

Les deux sequence diagrams (repris en annexe) implémentés par le prototype sont :

- *« init wrapper » qui aborde l'initialisation d'un nouveau wrapper auprès d'un médiateur*
- *« sequence analyse » qui détaille l'exécution d'une demande d'analyse d'un biologiste*

Les différentes étapes de la réalisation sont schématisée à la figure 25 ci-dessous

[17] http://www.iona.com/products/orbacus_home_over.htm

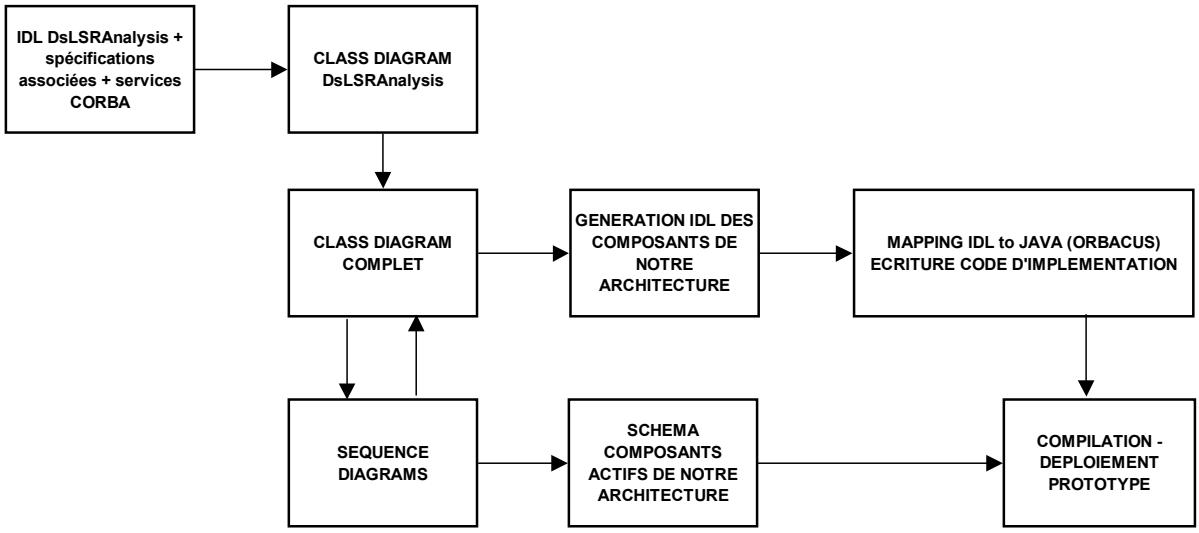

Figure 25 : Etapes suivies pour la réalisation du prototype

5.2 Schéma des composants actifs

Sur base des sequence diagrams, nous avons dérivé un schéma des composants actifs d'un wrapper et d'un médiateur (figure 26) dans lequel les composants distribués (en bleu) sont séparés des composants « locaux » ne devant pas être distribués (en gris).

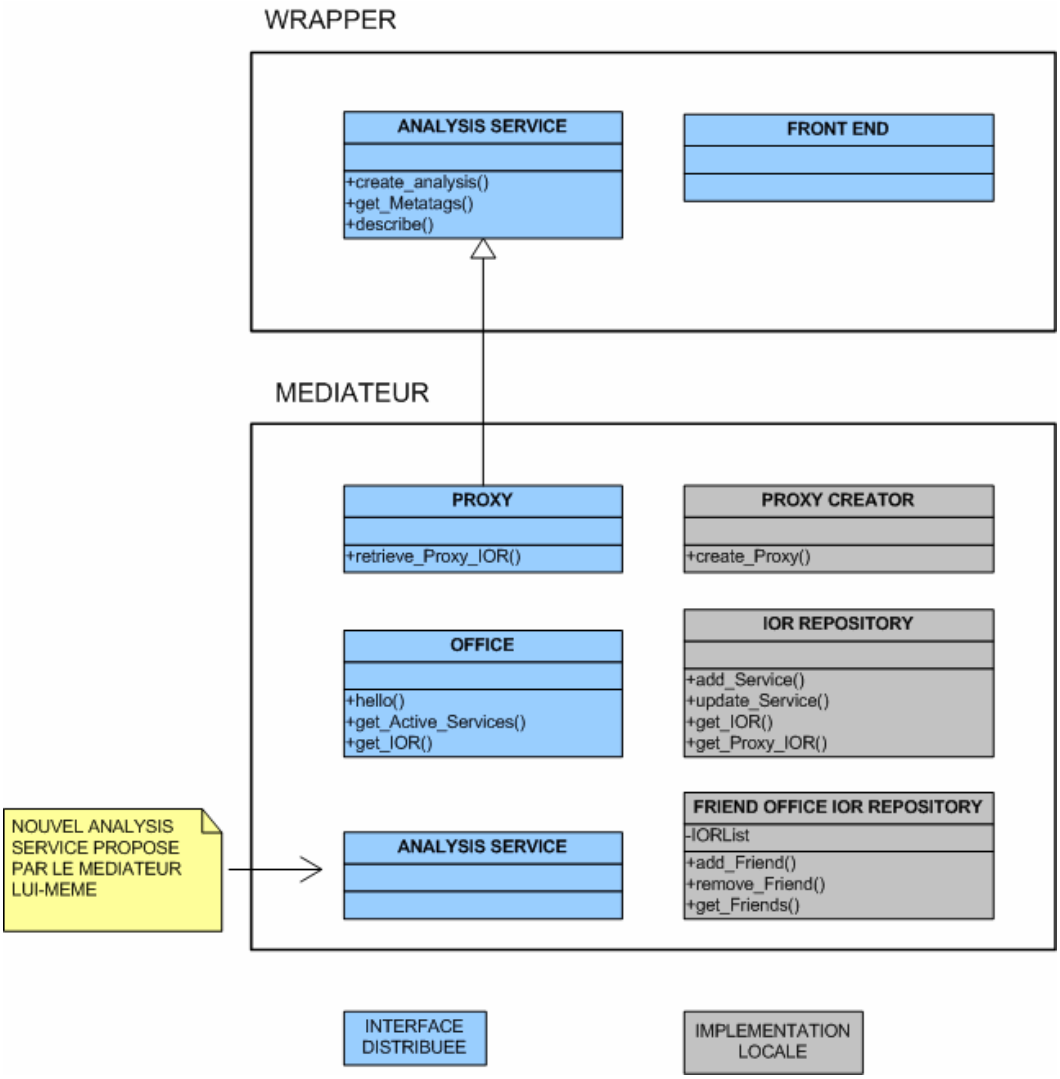

Figure 26 : Schéma des composants actifs

On observe dans ce schéma que l'interface proxy hérite de l'interface AnalysisService. Notons également le nouvel AnalysisService proposé par le médiateur lui-même, cet AnalysisService résultant de la composition de deux ou plusieurs Services (« script » faisant se succéder différents AnalysisServices où chacun prend en inputs les outputs du précédent, cf. aspect en couche du médiateur).

5.3 Constitution du class diagram du module DsLSRAnalysis

En ce qui concerne les composants distribués, nous sommes partis de la description IDL du module DsLSRAnalysis qui nécessite l'utilisation de description IDL d'autres modules:

orb.idl
CosPropertyService.idl
CosEventChannelAdmin.idl
CosLifeCycle.idl
TimeBase.idl

Nous nous sommes procuré ces fichiers sur le site officiel de l'OMG.

Afin d'obtenir automatiquement un class diagram dérivé de cet IDL nous avons utilisé le logiciel Together 6.0 de TogetherSoft[18]. Together est un outil CASE permettant la conception logicielle selon la norme UML (Unified Modeling Language) définie par l'OMG: use case, class diagram, sequence diagram… Il génère automatiquement le code associé au class diagram dans différents langages tels que IDL ou Java, mais il peut aussi générer le class diagram au départ du code. Cette génération fonctionne donc dans les deux sens en cours de projet. Nous avons ainsi une parfaite cohérence entre le code, les class et sequence diagrams.

Après avoir regroupé tous les fichiers IDL nécessaires dans un répertoire, nous avons généré avec Together le class diagram correspondant au module DsLSRAnalysis.

5.4 Ajout des éléments spécifiques à notre architecture

Il restait à compléter le class diagram en y ajoutant les nouvelles interfaces des composants distribués que notre architecture imposait, à savoir le Médiateur avec son Office et son Proxy, le Wrapper avec son Front end ainsi que l'interface correspondant au Client Les attributs et méthodes nécessaires on été rajoutés au sein de chacune des classes (figure 27).

[s] http://www.togethersoft.com/

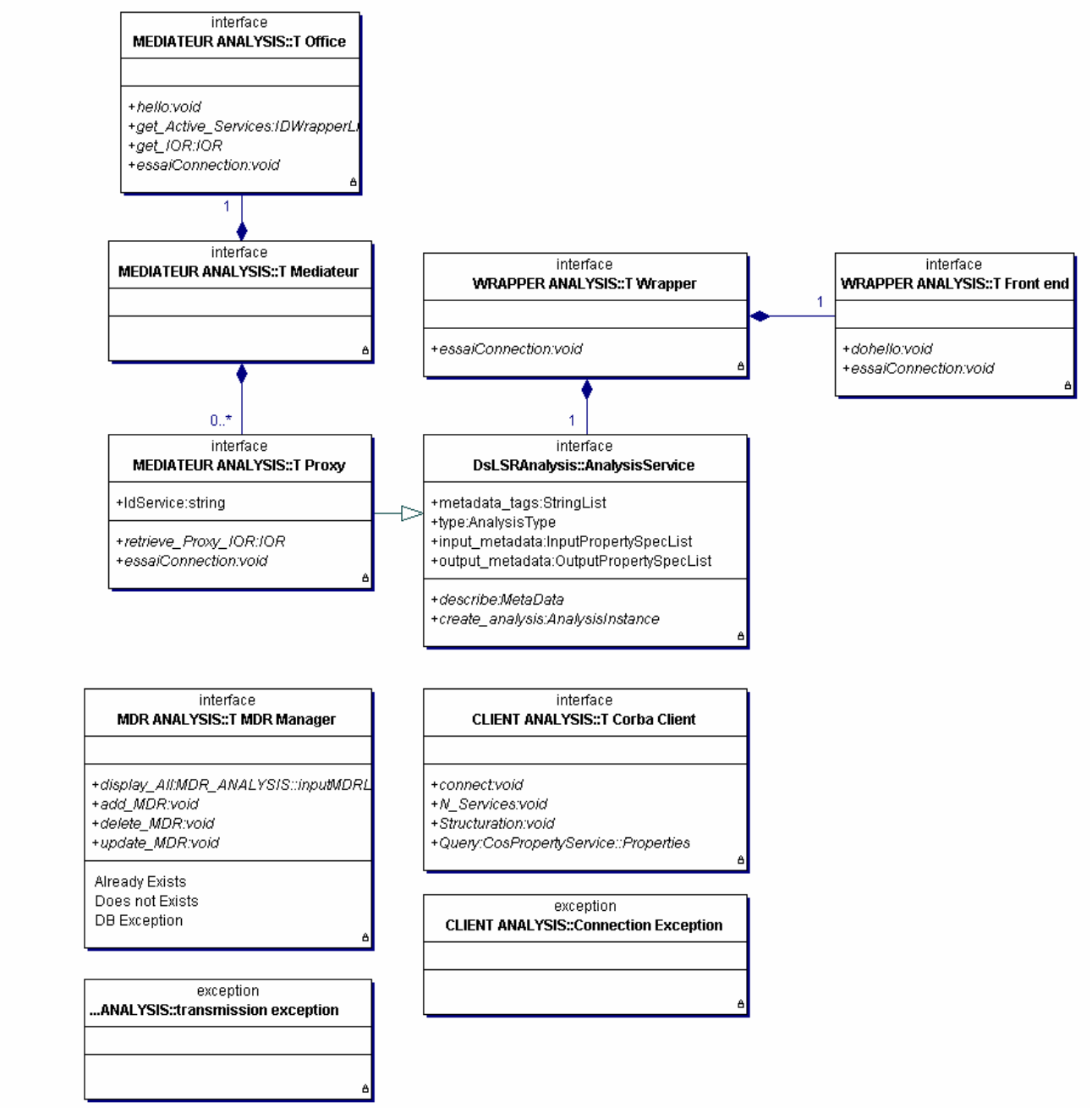

Figure 27 : Class diagram des éléments propres à notre architecture

Des nouveaux modules IDL ont été créés afin de contenir ces nouvelles interfaces. Chaque nouveau module donne lieu à un fichier IDL.
Il s'agit des fichiers

- *MDR_ANALYSIS.IDL*
- *CLIENT_ANALYSIS.IDL*
- *MEDIATEUR_ANALYSIS.IDL*
- *WRAPPER_ANALYSIS.IDL*.

5.5 Génération des amorces nécessaires à une application CORBA

Ayant la description IDL des composants distribués de notre application, il restait à compiler ce code afin de générer les amorces clientes et serveur. Cette compilation s'appelle le « **mapping** ». (Nous avons utilisé l'approche par héritage lors du mapping)

Les fichiers
DsLSRAnalysis.idl
CLIENT_ANALYSIS.idl
MEDIATEUR_ANALYSIS.idl
WRAPPER_ANALYSIS.idl

ont été compilés et donné lieu à tous les fichiers d'amorces regroupés dans les packages[19]

DsLSRAnalysis
CLIENT_ANALYSIS
MEDIATEUR_ANALYSIS
WRAPPER_ANALYSIS.

A titre d'exemple nous pouvons citer le résultat du mapping provenant du fichier *WRAPPER_ANALYSIS.idl* décrivant les deux interfaces *T_Front_end* et *T_Wrapper*[20]

Si on regarde les fichiers générés pour *T_Wrapper* nous découvrons notamment

- *T_WrapperOperations.java* : il s'agit d'une interface énumérant les opérations contenues dans l'interface *T_Wrapper*.
- *T_WrapperStub.java* : il s'agit de l'amorce client (stub) servant à émettre les requêtes.
- *T_Wrapper.java* : il s'agit de l'interface qui représente l'objet de type *T_Wrapper*. accessible à distance. Elle étend *T_WrapperOperations*
- *T_WrapperPOA.java* : il s'agit de la classe squelette servant à utiliser le POA.
- *T_WrapperHolder.java* : les classes Holder sont des classes conteneur utilisées lorsque l'on veut passer un objet de type utilisateur en argument, modifier la valeur de cet objet et récupérer cet objet modifié (problème inhérent à Java).

5.6 Ecriture des classes d'implémentation

A ce stade du projet, nous disposons pour chaque composant (Client, Médiateur et Wrapper) de ses interfaces et des classes générées lors du mapping. La situation pour le wrapper est illustrée par la figure 28.

[19] Package : Regroupement de classes.
[20] Toutes les interfaces non dérivées de la norme BSA ont un nom commençant par T_

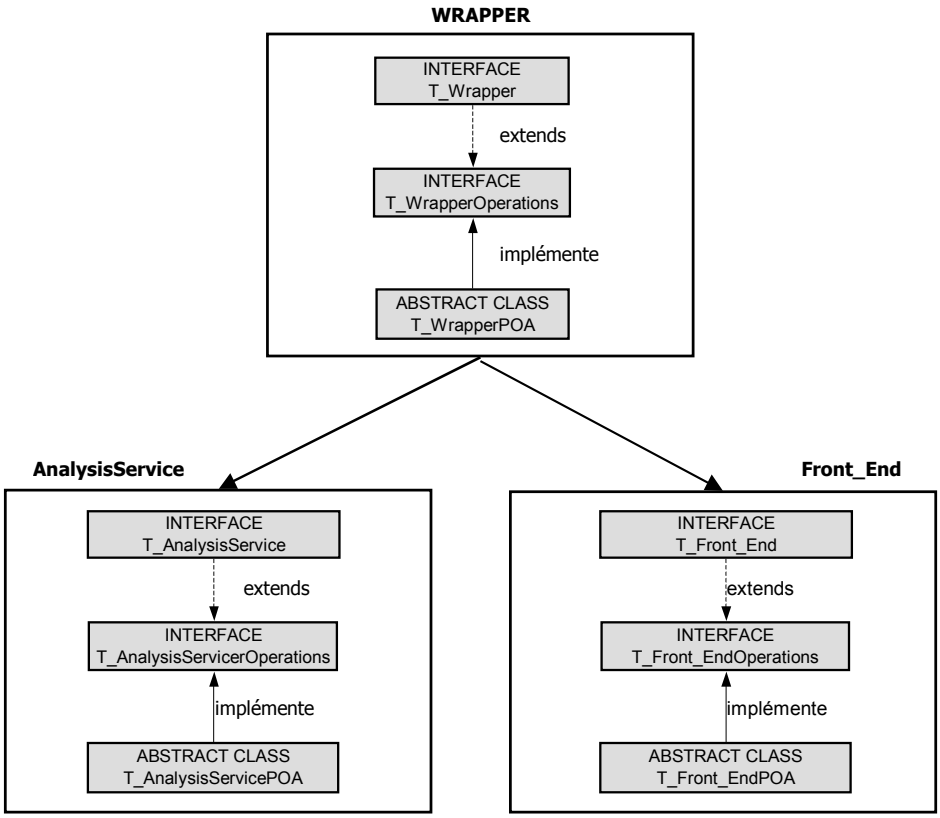

Figure 28 : Structure d'un wrapper (1)

Les classes XXX*POA* sont des classes abstraites qu'il est nécessaire d'implémenter. Ceci sera réalisé au travers des classes XXX*_Impl* (Figure 29)

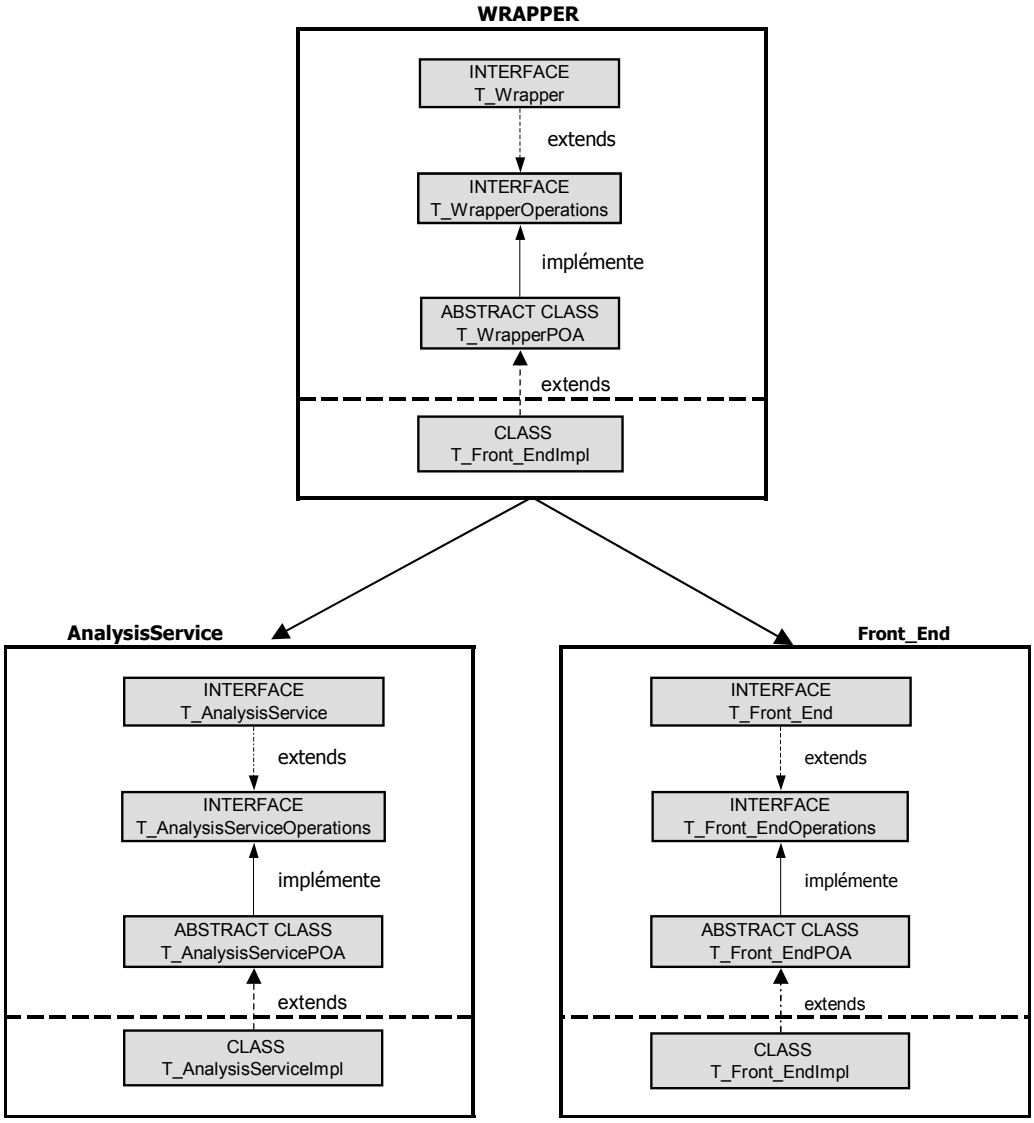

Figure 29 : Structure d'un wrapper (2)

Pour pouvoir exécuter les différents composants, nous avons créé quatre packages

CLIENT_ANALYSIS_SERVER
WRAPPER_ANALYSIS_SERVER
MEDIATEUR_ANALYSIS_SERVER
BIOLOGIST_SERVER

Ainsi par exemple, le package *WRAPPER_ANALYSIS_SERVER* contient la classe *WRAPPER_A_SERVER* renfermant la méthode *main()* nécessaire au démarrage du wrapper ainsi que les classes d'implémentation définies à la figure 29. Un wrapper contient donc les packages *WRAPPER_ANALYSIS* et *WRAPPER_ANALYSIS_SERVER* incluant toutes les classes nécessaires à son fonctionnement (figure 30).

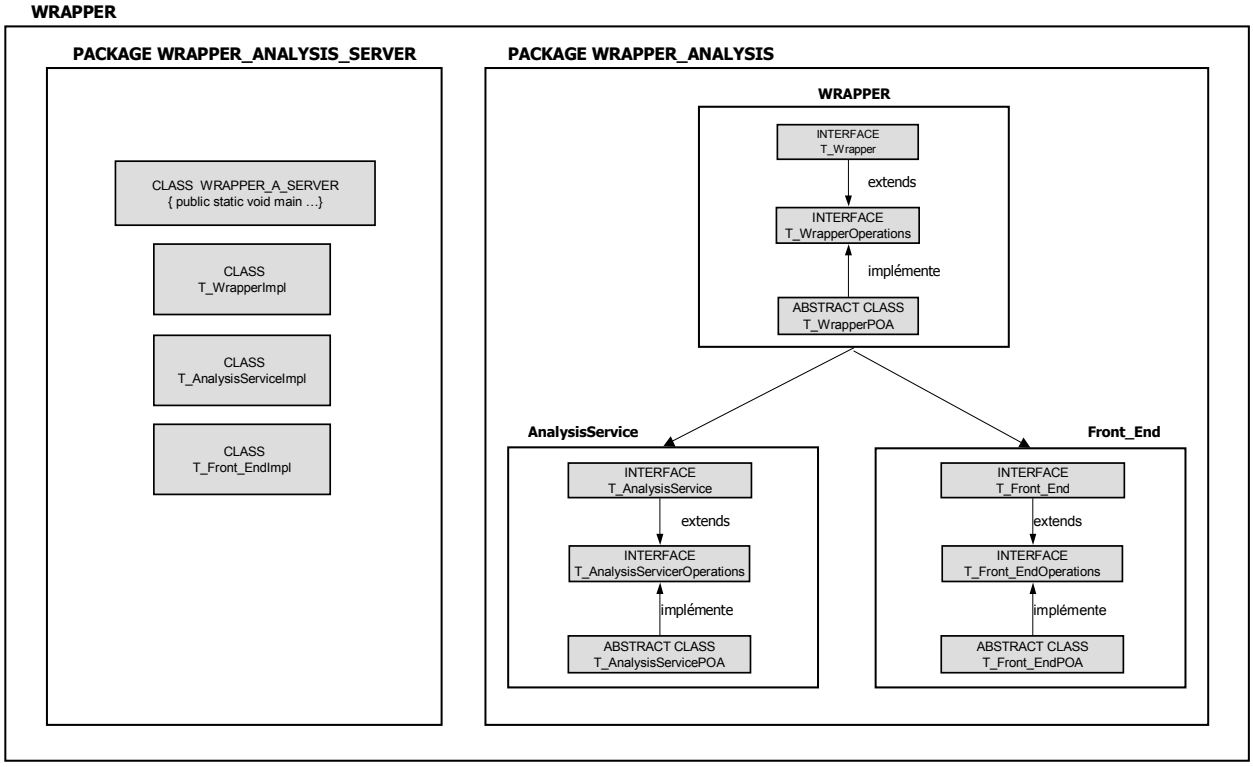

Figure 30 : Vue complète du déploiement d'un wrapper

Notons que dans le cas du médiateur qui possède des composants locaux non distribués (*proxy_creator*, *IOR_Repository* et *Fiend_OFFICE_IOR_REPOSITORY)* les classes nécessaires à l'implémentation se retrouveront dans le package *MEDIATEUR_ANALYSIS_SERVER*.

5.7 Quelques détails d'implémentation

Nous donnons ici à titre d'illustration quelques détails de notre implémentation. Le détail du code se trouve sur le CDROM (\BENPROJECT_java).

Les classes d'implémentation étendent les classes générées par le mapping (approche par héritage), par exemple, pour l'Office du médiateur:

```
« . . .     public class T_Office_Impl extends T_OfficePOA
      {
            public void hello(String inIDW,String W_IOR,String W_FE_IOR)
            throws transmission_exception
... ».
```

Implémentation de la méthode *create_analysis()*

Nous avons instancié un servant d'AnalysisInstance – AnalysisInstanceImpl –qui contiendra lui-même un objet de type AnalysisInstance

```
« . . .
public class AnalysisInstanceImpl extends AnalysisInstancePOA
   {
      DsLSRAnalysis.AnalysisInstance      theAI;
      . . .
      // Constructeur
      AnalysisInstanceImpl(String ids,CosPropertyService.Property[] inprop)
      . . .
      // AIimpl contient un AI qui pourra être invoqué (orb)
      // mécanisme générique serveur
      //([objet à invoquer] = [objetimpl]._this(orb)]

      AnalysisInstance aiMed = this._this(orb);
      this.theAI = aiMed;
      . . . »
```

Lorsque la méthode *create_analysis()* est appelée, elle instancie un AnalysisInstance et un Job Control. Le client doit alors invoquer les méthodes *run()* et *wait()* sur ce Job Control . Notre implémentation fonctionne de la manière suivante (figure 31) :

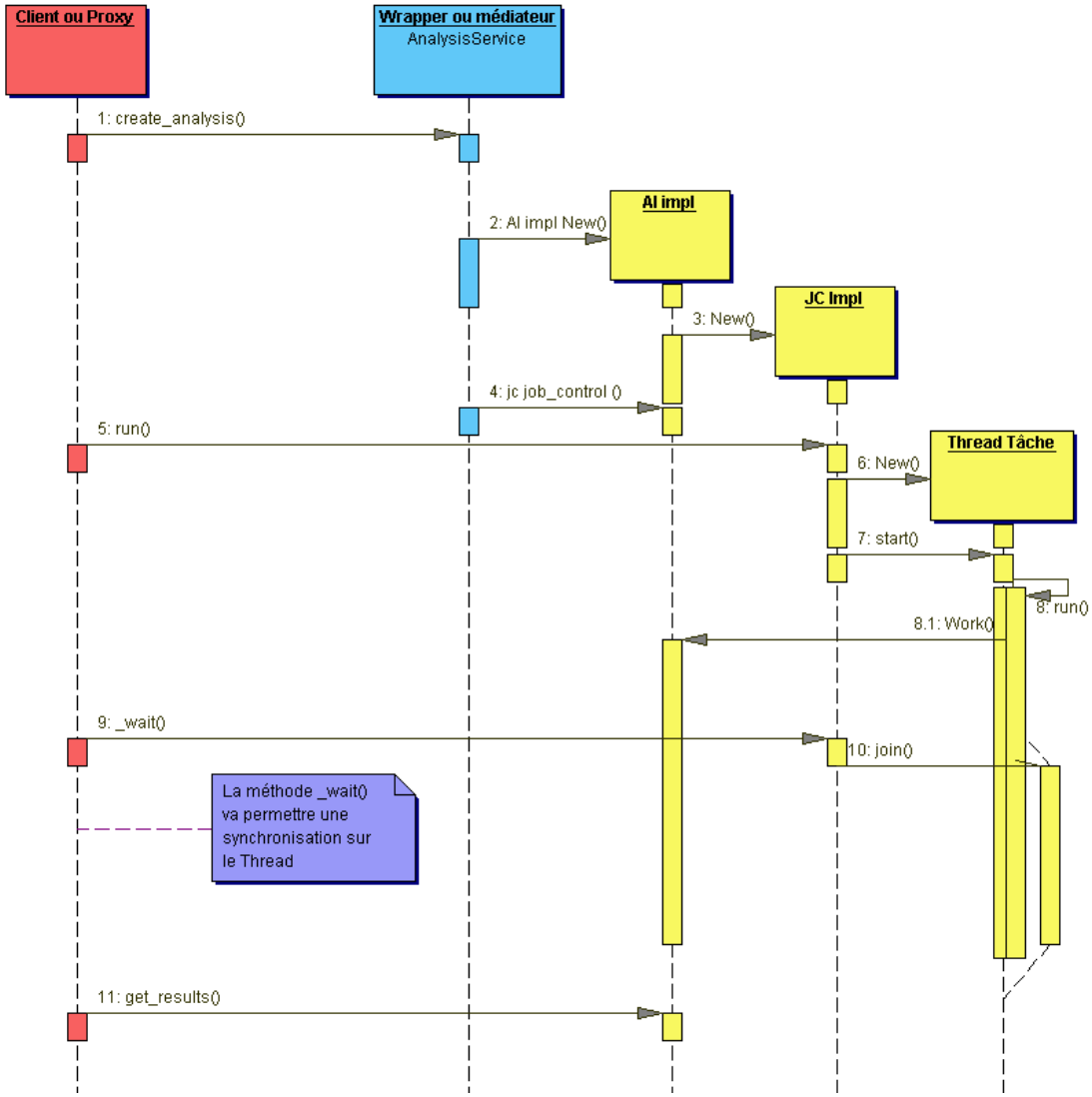

Figure 31 : Sequence diagram illustrant l'implémentation du *create_analysis()*

Un *run()* appliqué à un objet JobControl va générer un thread qui va démarrer la pseudo-analyse au sein de l'AnalysisInstance du proxy. En effet, dans le cas de notre architecture l'AnalysisInstance a pour seule fonction de propager le *create_analysis()* vers un wrapper. La méthode *wait()* appliquée au JobControl va se synchroniser sur le thread qui ne s'achèvera que lorsque le *create_analysis()* du wrapper aura terminé son travail et que les résultats auront été communiqués à l'AnalysisInstance du proxy. A ce moment le client corba peut à son tour récupérer le résultat de l'analyse et le renvoyer au biologiste.

5.8. Fonctionnement du prototype

Le déploiement du prototype est décrit sur le CDROM (\Fichiers d'auto-exécution\ReadMe.html). Ce prototype simule la déclaration d'un wrapper auprès d'un médiateur et la soumission d'une demande d'analyse au médiateur par un client. Le médiateur transmettra la demande au wrapper capable de la traiter. Finalement, les résultats seront transmis au client. Le tout en conformité avec la norme BSA.
Les captures d'écran ci-dessous (figures 32 à 36) illustrent le fonctionnement du prototype.

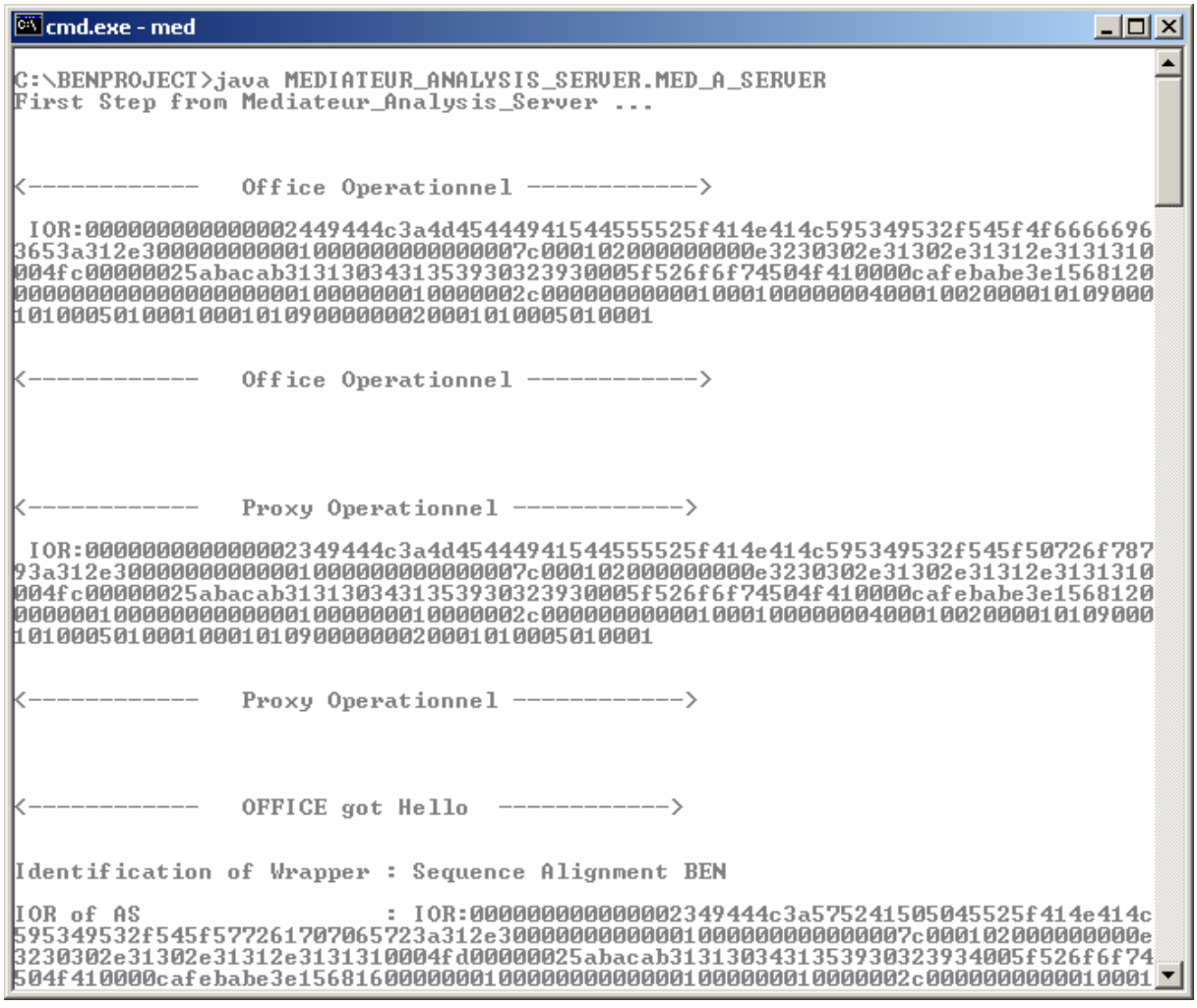

Figure 32 : Prototype – Vue du médiateur (1)

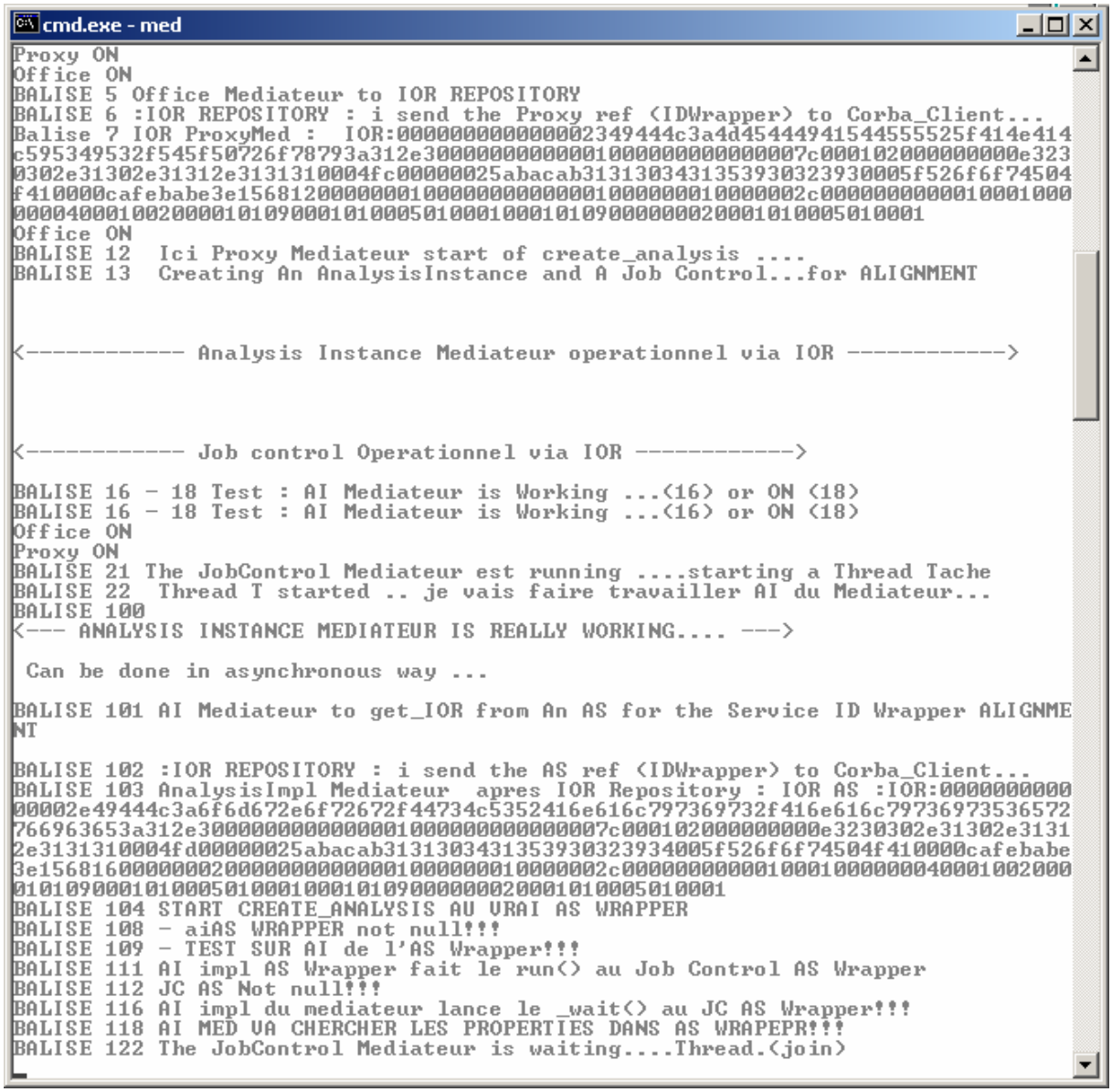

Figure 33 : Prototype – Vue du médiateur (2)

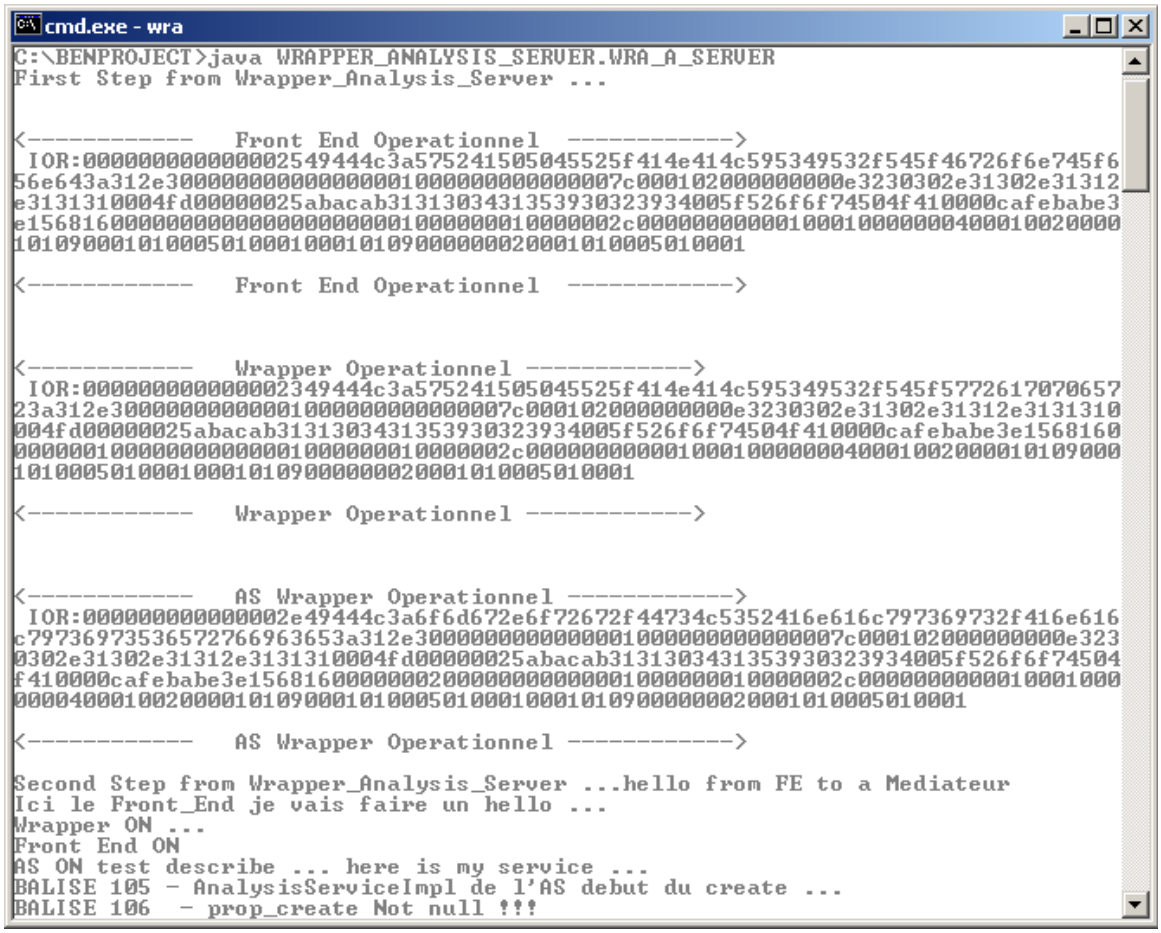

Figure 34 : Prototype – Vue du Wrapper

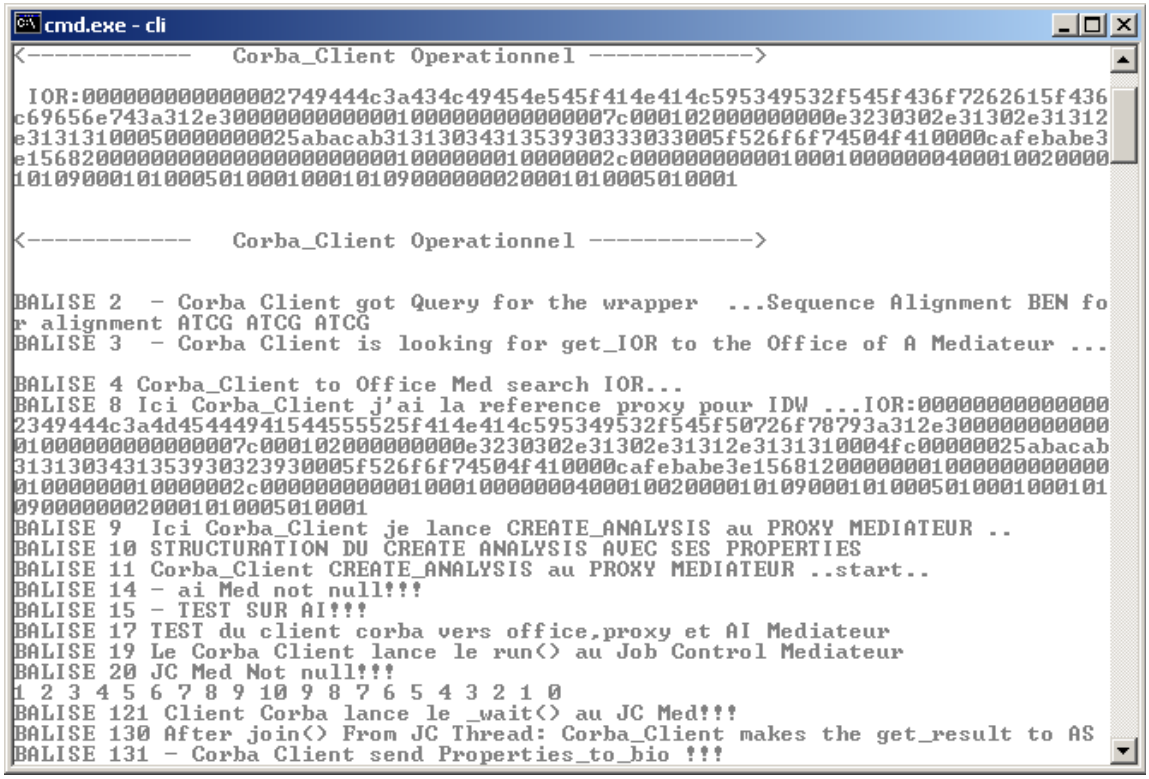

Figure 35 : Prototype – Vue du client corba

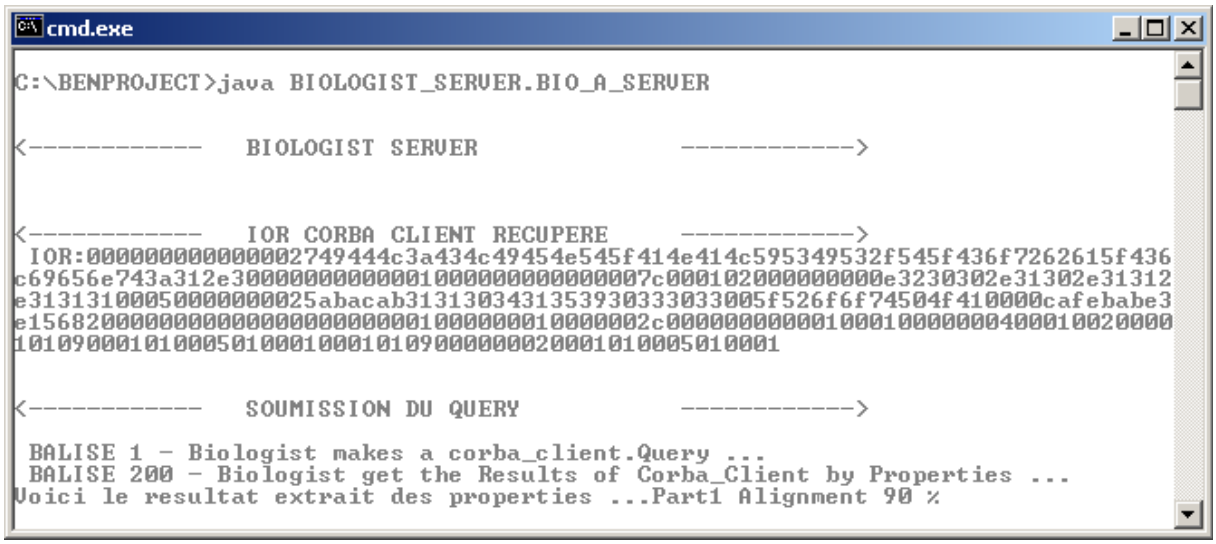

Figure 36 : Prototype – Vue du Biologiste

5.9. Conclusion

La réalisation du prototype nous a permis de vérifier notre analyse, plus précisément que le système élaboré pour la transmission d'une requête fonctionne effectivement.

Chapitre 6

Discussion

Différentes tentatives de distribution des ressources bioinformatiques

Comme nous l'avons déjà montré les principales limitations de la bioinformatique actuelle sont :

- la dispersion des données, dans des formats différents et avec des structurations différentes rendant difficile l'élaboration de requêtes complexes
- Le grand nombre et la grande hétérogénéité des logiciels d'analyse
- L'hétérogénéité des environnements (clients et serveurs)

Certaines équipes ont travaillé à l'intégration [Achard 00] et à l'échange [Martin 01] des données grâce au standard XML (eXtensible Markup Language). XML permet de créer des documents structurés à l'aide de balises. Il a pour avantage qu'il est aisé de parcourir le document ainsi formé pour en isoler l'information, que ce soit par voie logicielle ou visuelle par un humain. De même, de nouveaux éléments d'information peuvent facilement être ajoutés au document et enfin XML est orienté Internet. Par contre, à son débit, il est difficile de reproduire la sémantique complexe des données biologiques, d'autant que XML est limité quant aux types de données utilisables. De plus, il n'y a aucune méthode d'analyse associée directement au document. Enfin, il n'y a pas de mécanisme d'indexation ou de clustering de données. Il est donc nécessaire de parser tout le document pour en extraire l'information, ce qui entraîne de faibles performances.

XML ne répond donc que partiellement au problème de standardisation et de distribution des données et aucunement au problème de la distribution des outils d'analyse.

Les différentes normes publiées par l'OMG (CORBA et BSA) apportent des solutions globales et plus complètes. Rappelons que CORBA fournit des interfaces indépendantes des environnements et des modèles pour des applications orientées objet portables et distribuées. Son indépendance vis à vis des langages de programmation, des environnements et des protocoles réseaux le rend attractif pour le développement de nouvelles applications de distribution et d'analyse de données biologiques.

De récentes publications présentent le développement d'interfaces CORBA dans le contexte de la bioinformatique [Parsons 00] [Muilu 01]. Le développement de serveurs CORBA qui fournissent un accès aux données biologiques nécessite un modèle objet représentant ces données [Maltchenko 98]. Ce modèle peut alors être exprimé via une interface IDL. Ainsi, les données de l'EMBL sont distribuées et accessibles via des interfaces IDL développées par [Wang 00]. Gardons bien à l'esprit que l'utilisation de ces interfaces spécifiques nécessite l'écriture de clients capables d'utiliser ces interfaces.

La récente norme BSA constitue de ce fait une avancée importante dans la standardisation des interfaces IDL décrivant les données biologiques et le mécanisme d'analyse. A ce propos, il est important de noter que celui-ci est générique. Tout les outils d'analyse possèdent la même interface générique et toutes les demandes d'analyse se font via la méthode `create_analysis()` appliquée sur l'outil d'analyse ad hoc. Ce mécanisme a également

l'énorme avantage que les souches utilisées pour distribuer ces objets sur le bus CORBA sont génériques. Il n'est pas nécessaire de produire et distribuer aux clients des souches spécifiques lors de chaque ajout d'une analyse particulière au système. Vu le grand nombre des logiciels, et leur évolution continue, ceci constitue un élément important de l'architecture proposée.

Apports de notre architecture

Nous avons élaboré l'architecture d'un système fédéré dynamique d'utilisation de ressources bioinformatiques hétérogènes et ce, sur base de l'utilisation de la norme BSA et de CORBA.

Cette architecture est constituée d'éléments novateurs : le Master Definition Repository (MDR) et les médiateurs. Bien que nous n'ayons pas poursuivi le développement du concept du MDR, son rôle est essentiel car il standardise et centralise la définition des services offerts par le système.
Les médiateurs quant à eux possèdent plusieurs rôles importants

- Ils centralisent les services offerts par le système
- Il constituent des points d'accès pour les clients

Ces rôles seraient joués par une couche « de base » des médiateurs. Des couches plus élevées pourraient permettre la gestion des accès aux wrappers (accès protégé par mot de passe), la gestion de la sécurité, l'offre d'analyses complexes résultant de l'intégration de services offerts existants, etc.

Notre solution impose néanmoins pour sa mise en œuvre l'utilisation d'interfaces non décrites dans la norme BSA. Cependant, l'implémentation modulaire menée dans ce projet permet la réutilisation de nombreux modules génériques tels que l'Office des médiateurs ou le Front-end des wrappers. Notons que la norme BSA ne définit pas certaines responsabilités du client, et que quelle que soit l'architecture utilisée, il sera de toute manière indispensable d'utiliser des méthodes ou des interfaces non décrites.

Un apport important de notre architecture est la simplicité de sa dynamique. Nous avons mis sur pied un mécanisme totalement transparent de transmission des requêtes et des services offerts à travers le système. Cependant, plusieurs questions restent posées :

- Il y a lieu de s'assurer que lorsqu'un wrapper se déclare auprès d'un médiateur, cette déclaration soit répercutée à tous les médiateurs de la fédération et ce sans répétitions inutiles et sans problèmes de formation de boucle. Cela revient à construire un arbre de recouvrement minimum *(spanning tree*).
- Une requête adressée à un médiateur peut transiter par d'autres médiateurs pour parvenir au wrapper capable de l'exécuter. Le problème consiste alors à optimiser le chemin à parcourir en tenant compte de différents critères tels que le chemin le plus court, la charge du réseau, etc. Pour le résoudre on pourrait s'inspirer du fonctionnement de la couche réseau du modèle TCP/IP et envisager la construction de tables de routage.

Les médiateurs en centralisant les services offerts par le système permettent aux clients de les interroger afin d'en obtenir la liste continuellement mise à jour (via. notre méthode `get_Active_Services()`. Les applications clientes sont alors responsables de la structuration et de la présentation de ces services aux utilisateurs. La génération automatique de telles interfaces a notamment été réalisée par [Letondal 01] à partir de la description au format XML des fonctionnalités de programmes de biologie moléculaire et de leurs inputs et

outputs. Cette approche pourrait être utilisée pour la création d'interfaces en vue de l'utilisation de nos médiateurs.

Les tentatives d'implémentation de la norme BSA sont peu nombreuses à ce jour. Cependant, nos médiateurs pourront utiliser n'importe quel wrapper qui encapsule n'importe quelle base de données ou application existante pour autant que ce wrapper soit conforme à la norme BSA. De même toute application cliente développée afin de manipuler des objets BSA peut s'adresser directement aux médiateurs de la fédération. Notre architecture s'intégrera donc facilement dans de futurs travaux s'articulant autour de la norme BSA.

Les Services web constituent-ils une alternative sérieuse ?

Une technologie émergente très médiatisée, celle des Services Web[21], est présentée par certains comme étant LE moyen d'assurer l'interopérabilité entre les ressources informatiques. CORBA a parfois été décrié comme étant d'un abord plus ardu et posant des problèmes lors de l'utilisation de firewall.

Les Services Web sont une technologie qui permet à des applications de dialoguer à distance via Internet, et ceci indépendamment des plates-formes ou des langages sur lesquelles elles reposent. Ils nécessitent l'utilisation de protocoles afin de standardiser les modes d'invocation des composants applicatifs, il s'agit de SOAP (Simple Object Access Protocol) et WSDL (Web Service Definition Language). Le protocole SOAP définit la structure des messages échangés par les applications via Internet tandis que WSDL fournit un mode de description des composants applicatifs permettant les invocations distantes par échange de messages au format SOAP. Le référencement des Services Web déployés peut s'effectuer à l'aide d'un annuaire, par exemple UDDI (Universal Description, Discovery an Integration), ce qui permet aux Services Web de s'invoquer dynamiquement.

Les Sevices Web doivent être décrits avec par exemple WSDL, ce qui est fait par l'IDL avec CORBA. L'échange des messages standardisé par SOAP est pris en charge avec CORBA par l'ORB, et l'identification et la localisation des Services Web avec UDDI par exemple est organisée par le Naming Service de CORBA.. Les fonctionnalités offertes par les services web ne constituent donc pas une véritable révolution par rapport à CORBA. Le but à atteindre l'interopérabilité entre ressources informatiques à travers un réseau et indépendamment de leur plates-formes d'origine est plus ou moins identique, mais les technologies mises en oeuvres par les services web veulent utiliser des standards de données plus « orientés » Internet.

Dans le cas qui nous préoccupe, celui de l'utilisation de ressources bioinformatiques, les Services Web constituent ils une nouvelle piste à explorer ? Nous avons déjà discuté de la difficulté de modéliser les données biologiques et des problèmes liés aux différents formats des données. Les protocoles d'échange comme SOAP s'appuyant sur XML, ne sont actuellement pas adéquats pour l'échange des données complexes manipulées par la bioinformatique.

Les Services Web imposent également de standardiser les données, de décrire les processus métier dans le format requis pour l'utilisation de cette technologie. Aucun standard

[21] http://www.w3.org

actuellement ne s'est encore dégagé. L'utilisation des Services Web dans le domaine de la bioinformatique a déjà été tentée avec par exemple la distribution des données de l'EMBL[22]. La multiplication de ces services imposera la création d'un registry qui, à l'instar de notre MDR, les centralisera sous une forme standardisée et permettra ainsi leur accès. Ceci a été présenté par [Stein 02] qui propose une code de conduite destiné aux développeurs et utilisateurs de ressources bioinformatiques.

Nous pensons que les Sevices Web offrent des perspectives intéressantes, mais souffrent pour le moment de la multiplicité (sur fond de guerre commerciale) des différents standards proposés.

Nous avons vu que la norme BSA et CORBA ne souffrent pas de ces problèmes. La technologie CORBA bénéficie d'une excellente réputation au niveau de l'interopérabilité, de la portabilité, de la fiabilité, et fournit un grand nombre services (tolérance aux pannes, persistance, sécurité...). Enfin, la norme BSA suit un modèle orienté objet particulièrement bien adapté aux types d'objets complexes rencontrés en biologie.

Il est donc prématuré de croire que les services web constituent la solution finale. Non seulement cette technologie est encore trop récente, mais surtout l'élaboration et la promotion des standards est seulement en cours et actuellement de manière chaotique.

[22] http://www.ebi.ac.uk/xembl/

Conclusion

L'usage de la bioinformatique se révèle actuellement trop complexe pour l'utilisateur non initié à cette discipline de part le manque de standardisation des données, leur évolution, leur distribution, la diversité des logiciels d'analyse, l'hétérogénéité des environnements,.. Cela freine l'utilisation des différentes ressources, données biologiques ou systèmes d'analyse de ces données par des biologistes peu formés à cette tâche. Il existe de la part de la communauté scientifique une demande pressante de simplification et d'intégration de ces ressources.

De manière générale, CORBA permet l'intégration de ressources hétérogènes en fournissant une couche logicielle entre les clients et les services, services pouvant être localisés sur l'Internet ou sur un Intranet. Partant de l'utilisation de l'architecture CORBA et de la récente norme BSA définie également par l'OMG et, nous avons élaboré un système fédéré et dynamique d'utilisation de ressources bioinformatiques hétérogènes.

Une des clés de ce système est la création des médiateurs. Ces médiateurs sont des composants nouveaux chargés entre autre de gérer la transparence des accès au système que ce soit lors de la déclaration d'un wrapper ou d'une demande d'analyse. Ainsi un client peut adresser une demande d'analyse à un médiateur qui se chargera de la faire réaliser par le système.

Un autre point marquant de ce travail a été l'étude de la dynamique du système. Nous avons défini comment de nouveaux services sont mis à disposition du système, comment cette information se transmet à tous les médiateurs et comment une demande adressée à un médiateur sera transmise au travers du système jusqu'au fournisseur de services.

De plus, cette architecture est modulaire, réutilisable et expansible. Modulaire et réutilisable car de nombreux composants sont génériques donc utilisables par tous. Expansible car notre solution permet son développement « en couches ». Nous fournissons un service minimal que chacun est libre de compléter. Notre architecture, respectant la norme BSA, permet son intégration avec d'autres éléments (clients, wrappers ...) conformes à cette norme.

Grâce à cette étude et au prototype développé nous pensons avoir démontré que les efforts de standardisation, d'intégration des données et d'interopérabilité des applications ne sont pas vains et conduiront à une utilisation simplifiée et optimisée de la bioinformatique.

Références

Anonyme, « Distribution release notes », NCBI-GenBank Flat File Release 126.0, October 15, 2001

Achard F., Vaysseix G. and Barillot E., « XML, bioinformatics and data integration », *Bioinformatics*, 2001, Vol. 17, No. 2, p 115-125.

Brown, K. « La lecture du génome humain », 2000, *Pour la science*, No 275, p. 40-45.

Coupaye T., « Wrapping SRS with CORBA : from textual data to distributed objetcs », *Bioinformatics*, 1999, Vol. 15, No 4, p333-338.

Daniel J., *Au cœur de CORBA*, VUIBERT, Paris, France, 2000.

Geib J-M., Gransart C., Merle P., *Corba : Des concepts à la pratique*, DUNOD, Paris, France, 1999.

Gibas C., Jambeck P., *Developping bioinformatics computer skills,* O'REILLY, Sebastopol, USA, 2001.

Letondal C., « A web interface generator for molecular biology programs in Unix », *Bioinformatics*, 2001, Vol. 17, No.1, p. 73-82.

Maltchenko S., « The Bio-Objects project. Part 1 : The object data model core elements », *Bioinformatics*, 1998, Vol. 14, No. 6, p. 479-485.

Martin A., « Can we integrate bioinformatics data on the Internet ? », *TRENDS in Biotechnology*, 2001, Vol. 19, No. 9, p. 327-328.

Muilu J., Rodriguez-Tomé P., and Robinson A., « GBuilder-An application for the visualisation and integration of EST cluster data », *Genome Research*, 2001, Vol. 11, p 179-184.

Parsons J. D. and Rodriguez-Tomé P., « JESAM : CORBA software components to create and publish EST alignments and clusters », *Bioinformatics*, 2000, Vol., No 4, 16, p 315-325.

Stein L., Creating a bioinformatics nation, *Nature*, 2002, Vol. 417, p. 119-120.

Wang L., Rodriguez-Tomé P., Redaschi N., McNeil P., Robinson A. and Lijnzaad P., « Accessing and distributing EMBL data using CORBA (common object request broker architecture) », *Genome Biology*, 2000, Vol 1, No. 5**:**research0010.1-0010.10.

Zdobnov EM, Lopez R, Apweiler R, Etzold T. « The EBI SRS server-recent developments », *Bioinformatics*, 2002, Vol. 2, No 8, p. 368-373.

Ressources WWW

GenBank : http://www.ncbi.nih.gov/Genbank/GenbankOverview.html
European Molecular Biology Laboratory Nucleotide Sequence Database : http://www.ebi.ac.uk/embl/
DNA Data Bank of Japan : http://www.ddbj.nig.ac.jp/
Sequence Retrieval System : http://srs.ebi.ac.uk/
European Molecular Biology Open Software Suite : http://www.hgmp.mrc.ac.uk/Software/EMBOSS/
Belgian EMBnet Node ; http://ben.vub.ac.be/
EMBnet : http://www.embnet.org/
Common Object Request Broker Architecture : http://www.corba.org
Object Management Group : http://www.omg.org
Life Science Research Task Force : http://www.omg.org/homepages/lsr/
Biomolecular Sequence Analysis : http://www.omg.org/technology/documents/formal/biomolecular_sequence.htm
Together 6.0 : http://www.togethersoft.com/
ORBacus 4.1.0 : http://www.iona.com/products/orbacus_home_over.htm
Worl Wide Web Consortium : http://www.w3.org
XEMBL Project : http://www.ebi.ac.uk/xembl/

Contenu du CDROM

- BENPROJECT : fichiers compilés du prototype
- BENPROJECT_JAVA : fichiers sources du prototype
- Fichiers d'auto-exécution : fichiers.bat d'exécution du prototype et ReadMe.html
- Fichiers IDL : code IDL nécessaire au prototype
- IMAGES
  - CLASS DIAGRAMS : DsLSRAnalysis complété
    DsLSRBioObjects
  - SEQUENCE DIAGRAMS : Initialisation d'un wrapper
    Sequence Analyse
- NORMES OMG : Biomolecular Sequence Analysis Specification version 1.0
- OOC : Fichiers nécessaires à installer Orbacus
- TEXTE MEMOIRE

Annexes

- Annexe 1 Sequence diagram Init Wrapper
- Annexe 2 Sequence diagram Sequence Analysis