Cahier des charges du projet OptimEISTI

Client : EISTI

Prestataire : groupe d'étudiants ING2 : GI

Sommaire

Expression du besoin

Présentation générale

Le client EISTI est représenté par

- Roberto Bonato pour le site de Pau
- Bernard Glonneau, Rachid Chelouah et Hervé de Milleville pour le site de Cergy

Le prestataire est représenté par un groupe de 3 ou 4 étudiants d'ING2 inscrits dans la spécialité GI.

Nous, client EISTI, désirons acquérir un logiciel d'optimisation de problèmes non linéaires. Ce logiciel pourra être utilisé par l'EISTI à des fins pédagogiques. Un utilisateur doit pouvoir faire tourner les méthodes suivantes :

- Méthodes déterministes
  - méthode du gradient à pas fixe
  - méthode du gradient à pas optimal
  - méthode de Newton
  - méthode de projection
  - méthode de Lagrange
  - méthode de pénalisation
- Méthodes non déterministes
  - recuit simulé et variantes
  - algorithmes évolutionnaires
  - algorithme de recherche tabou
  - algorithmes de colonies de fourmis
  - algorithme pas essaim particulaire

La liste des méthodes peut être éventuellement revue entre le client et le prestataire.

Fonctionnalités principales

Ce logiciel doit impérativement comporter trois fonctionnalités principales :

1. **Constituer une base de données de problèmes** : Possibilité de créer, modifier, consulter et supprimer des problèmes d'optimisation non linéaire.
   Créer un problème d'optimisation non linéaire consiste à saisir sous une forme adéquate :
   - la fonction objectif
   - les contraintes d'inégalité
   - les contraintes d'égalité
2. **Exécuter un problème et restituer les résultats** :
   - sélectionner dans la base de problème un problème existant
   - choisir une méthode ou plusieurs méthodes de résolution
   - demander l'exécution et la restitution des résultats
   - éventuellement stocker les résultats dans une base adéquate pour de futures études sur ces dits résultats
3. **Faire des benchmarks sur les méthodes** :
   - sélectionner un certain nombre de méthodes pour comparer leurs performances
   - demander à la machine de choisir dans la base de problèmes au hasard (ou pas) un ou plusieurs problèmes et d'exécuter les méthodes sur ces problèmes et de restituer un tableau comparatif sur les performances

Fonctionnalités secondaires

Nous désirons en plus de ces fonctionnalités avoir d'autres fonctionnalités secondaires mais néanmoins importantes.

1. Créer des utilisateurs (avec un login/password)

2. Créer des groupes d'utilisateurs.
    - un utilisateur quelconque doit obligatoirement faire partie d'un groupe. Il peut être membre de plusieurs groupes
    - Chaque groupe d'utilisateurs gère son ensemble de problèmes d'optimisation. Deux utilisateurs d'un même groupe peuvent donc accéder aux trois fonctionnalités principales sur les mêmes problèmes et au même moment. Par contre un utilisateur n'a aucun droit sur les problèmes définis par un groupe dont il n'est pas membre.
- Importer des problèmes XML
  On pourra importer des problèmes à partir de fichiers XML dans la base de données
- Créer des documents imprimables
  L'utilisateur doit pouvoir générer trois types de documents imprimables
    - un document contenant l'énoncé d'un problème d'optimisation (voir de plusieurs)
    - un document contenant l'énoncé et la solution d'un problème après exécution de certaines méthodes (voir de plusieurs problèmes).
    - un document contenant les résultats d'un benchmark entre plusieurs méthodes.

Ce qui est précisé ci-dessus sous entend que les différents problèmes sont définis et donc associés à un groupe d'utilisateur.
De même, il faut pouvoir paramétrer le logiciel pour définir un administrateur qui seul aurait la possibilité de créer et supprimer des utilisateurs et des groupes. Quand un groupe est créé, l'administrateur doit inclure dans le groupe un ou plusieurs utilisateurs qui seront responsables du groupe. Un responsable de groupe a la possibilité d'ajouter ou de retirer des membres de son groupe.

Contraintes techniques et développement

Architecture

L'architecture du logiciel est une architecture 3 tiers (voir schéma ci-dessous) :

1) les clients
2) le serveur de requêtes (traiter les requêtes des clients et exécuter les calculs)
3) le serveur de bases de données

L'interface Homme/Machine du client

C'est un client lourd écrit en java en utilisant les package awt et swing.

SGBD

Le SGBD doit être impérativement Oracle. Les traitements spécifiques à la base de données devront être stockés dans la base de données et exécutés par le serveur de bases de données. On utilisera donc PL/SQL comme langage de développement pour ces traitements. Le client EISTI sera sensible à une répartition judicieuse (optimisation des charges) des traitements entre le serveur de requêtes et le serveur de bases de données.

Le serveur de requêtes

C'est une machine UNIX. Tous les traitements sur ce serveur doivent être développés en langage C avec éventuellement en plus des scripts. Les développements sur ce serveur doivent parfaitement intégrer l'aspect multi-utilisateurs en utilisant à bon escient le multi-processing et/ou le multi-threading.

Communication entre un client et le serveur de requêtes

On utilisera les protocoles TCP/IP et UDP.

Communication entre le serveur de requêtes

On utilisera le middleware ODBC pour que le serveur de requêtes communique avec le serveur de base de données.

Tests intermédiaires

Lors d'étapes intermédiaires du projet le client EISTI souhaite pouvoir tester :

- la logique de dialogue entre le ou les clients d'une part et le serveur de requêtes d'autre part sans que le serveur de bases de données fonctionne.
- La logique de dialogue entre le serveur de requêtes d'une part et le serveur de bases de données d'autre part sans passer par un client

Dans les deux cas de figures, le prestataire devra donc fournir des jeux de tests validés par le client EISTI.

Représentation des données dans la base

Les informations relatives aux utilisateurs et aux groupes d'utilisateurs sont modélisés et donc stockés en relationnel.
Les informations relatives aux problèmes et aux résultats après exécution doivent être modélisés et donc stocker en utilisant un mixte entre une représentation relationnelle et représentation XML. La représentation d'un problème doit être un arbre XML. La partie purement relationnelle ne doit concerner que l'identification du problème et les liens avec les groupes d'utilisateurs et les résultats associés à la résolution du dit problème. Il en est de même pour les résultats.

Organisation

Après lecture de ce cahier des charges, le prestataire s'engage à fournir au client comme premier travail, une définition précise du déroulement du projet. Cette définition doit contenir :

1. le découpage en tâches : chaque tâche doit contenir une définition claire et concise ainsi que (s'il y a lieu) des libellés des tâches qui doivent être impérativement effectuées avant.
2. pour chaque tâche, les personnes affectées à la dite tâche
3. les différents jalons (ou livrables)
4. un planning de déroulement du projet (en utilisant impérativement MS Project)

Ce premier travail devra être validé par le client EISTI avant de poursuivre le projet.

A chaque jalon, le prestataire doit fournir le livrable associé ainsi qu'un comparatif entre le planning effectif et le planning prévisionnel. Tout écart devra être expliqué.

Le seul site reconnu par le client pour déposer les livrables est AREL à un endroit convenu par le client EISTI.

A la fin de la mission, chaque personne du groupe prestataire devra évaluer ces collègues en respectant des règles qui seront fournies par l'équipe pédagogique.