CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

METHODES DE PROGRAMMATION SYSTEMES

UE NSY103

NANCY – METZ

PROJET
Année 2007 – 2008, deuxième semestre
Coefficient : 1/3

Travail à réaliser au plus tard pour le lundi 16/06/2008 17h00

Travail à rendre en version électronique :

- **au surveillant au début de l'examen final (avant la distribution du sujet),**
- **ou à envoyer avant l'examen à E. Desvigne : emmanuel@desvigne.org**

Création d'un middleware de système d'information de production

Les publicitaires l'ont compris et ont déjà utilisé l'argument dans leurs campagnes : un des grands enjeux des entreprises de production modernes est de pouvoir répondre aux attentes du client avec une réactivité la plus proche du “temps réel”. Le client d'un concessionnaire automobile fait l'acquisition d'une voiture rouge avec l'option “toit ouvrant” et “autoradio CD/MP3” ? L'information doit être répercutée dans les minutes ou les heures qui suivent dans le système d'information de production, afin que la construction du véhicule commence au plus tôt. Le client change d'avis sur la couleur du véhicule ? Le concessionnaire doit pouvoir savoir si la carrosserie est déjà entrée dans l'atelier de peinture de l'usine, et si ça n'est pas le cas, il doit pouvoir indiquer le changement d'option, afin qu'il soit pris en compte par les robots peintres. Inversement, certains surplus ou rationalisations de la chaîne de production permettent des économies ponctuelles sur certaines options. Cette information doit être relayée au réseau commercial, afin que les vendeurs puissent proposer des remises et faire la promotion de ces options.

Plusieurs méthodes permettent cette communication entre le système d'information de l'usine de production et le réseau de revendeurs :

- historiquement, un des premiers systèmes mis en place était la communication par “boîtes aux lettres”. Classiquement, les informations d'un composant ayant de l'information à émettre sont formatées dans un fichier (une ligne par entrée, avec un format “positionnel”, ou champs séparés par un caractère séparateur), qui est déposé sur un serveur de fichiers (via FTP, ou par e-mail classique, ou par messagerie électronique sécurisée, etc.). Et chaque composant se connecte régulièrement à son espace sur le serveur de fichiers (par exemple, tous les ¼ d'heure), pour y relever les informations le concernant. Evidemment, lorsqu'un composant du système se connecte au serveur de fichiers, et s'il constate la présence d'un fichier le concernant, il n'a aucun moyen de savoir à priori s'il peut le récupérer, ou si ce dernier est en cours de remplissage (et par conséquent, dans un état incomplet/incohérent). C'est pourquoi, en général, le système de boîtes aux lettres met les données dans un fichier “xxx.dat”. Et une fois ce fichier remplit et correctement terminé, un fichier vide ayant comme nom “xxx.ok” est créé. Le destinataire ne récupère donc que les fichiers “*.dat” pour lesquels un fichier “*.ok” a été généré. Cette solution, simple à mettre en oeuvre, n'est pas optimale pour un service “temps réel” ;
- une autre solution consiste à créer un lien socket bi-directionnel entre deux composants ayant à s'échanger régulièrement de l'information. Si cette solution offre une réponse à la problématique “communication temps-réel”, elle pose d'autres problèmes :
    - chaque composante doit surveiller la connexion réseau, détecter une éventuelle coupure, et pouvoir se reconnecter et reprendre le fil des transactions une fois le réseau rétabli. Ceci suppose la mise en oeuvre d'algorithmes complexes à réaliser ;
    - de plus, le dialogue machine<->machine se fait souvent dans un protocole propriétaire. Ce qui rend le système d'information dépendant du fournisseur de chaque composante. Changer une composante du système d'information demande soit le développement du même protocole propriétaire que celui qui était utilisé précédemment, soit une refonte complète de l'interface ;
- enfin, une solution moderne commence à émerger ces dernières années : la solution de “bus applicatif”. Chaque composante dialogue avec une autre composante sous la forme d'une requête, à la façon des “remote procedure call” (appels de fonction à distance). Ces dernières années, pour normaliser le plus possible les échanges, la mode est d'utiliser les standards de fait d'Internet : échanges des données en les formatant en XML, et utilisation des “web services” (les appels fonction à distance utilisent les mêmes mécanismes que les requêtes web d'internet).

Or, dans la réalité des systèmes d'information de production, il n'est pas rare de se retrouver avec des logiciels d'architectures étérogènes. Certains systèmes très anciens ne fonctionnent que par un dialogue de type “boîtes aux lettres”. D'autres supportent les connexions sockets. Enfin, les plus modernes savent répondre à des requêtes de type webservices.

Le défi consiste à interfacer le système d'information de l'usine de production avec les systèmes d'information de tous les acteurs du réseau commercial, quelle que soit la technologie que ceux-ci supportent. Ce travail est accompli par un “**middleware**”.

Spécifications du middleware à réaliser

Le middleware, qui tourne sur un serveur dédié, devra être capable d'adapter (aussi bien sur le fond que sur la forme) le mécanisme de communication bidirectionnel de chaque système informatique de chaque concessionnaire avec celui du système d'information de l'usine de production.

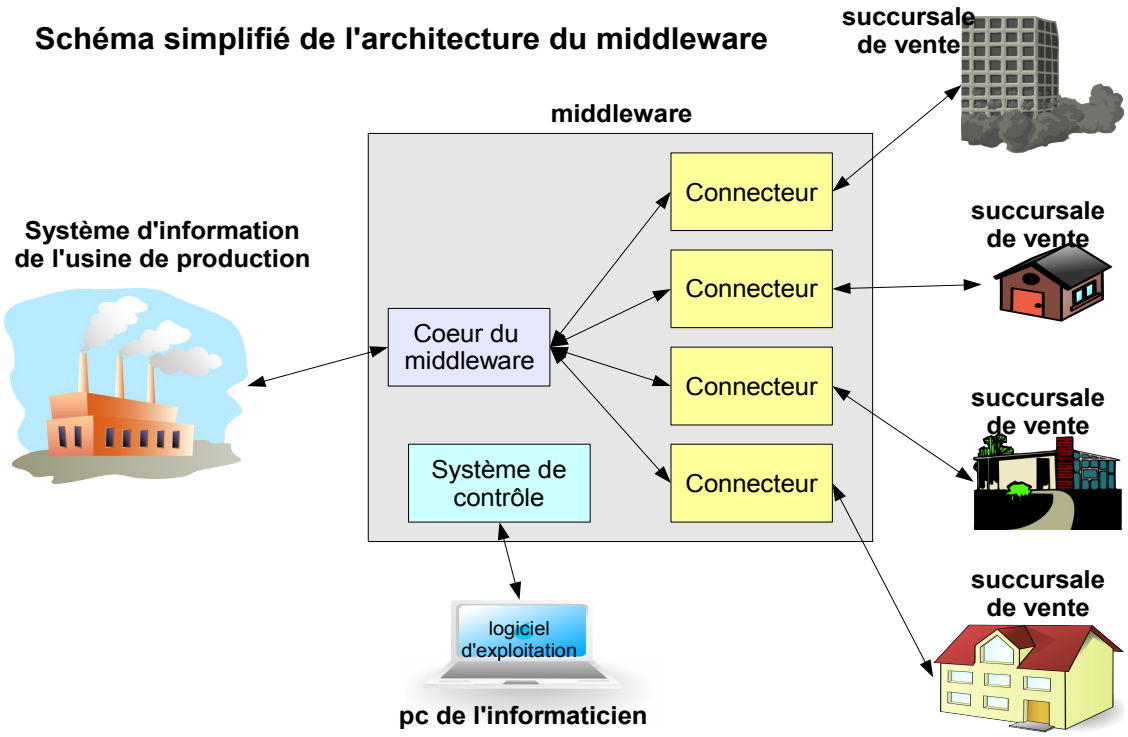
Pour ce, le coeur du middleware sera continuellement connecté de façon bidirectionnelle avec le système d'information de l'usine de production (lien socket). Ensuite, un certain nombre de “connecteurs” seront lancés. Il y aura autant de connecteurs que de succursales avec lesquelles il faut communiquer. C'est le rôle de chaque connecteur d'adapter la communication inter-entreprise, aussi bien sur le fond (transformation d'un fichier texte positionnel en XML par exemple) que sur la forme (communication par boîtes aux lettres, connexion socket, ou par webservices). Chaque connecteur dialogue en direct avec le coeur du middleware pour communiquer avec le système d'information de l'usine de production. Un connecteur peut être créé ou détruit “à la volée”, sans avoir à redémarrer tout le middleware. Un connecteur est un élément multitâche. Par exemple, il pourra très bien être en train de dialoguer avec le coeur du middleware, et recevoir simultanément une requête de la part du concessionnaire auquel il est rattaché.

Chaque connecteur, quel que soit son type, gère en temps réel des données le concernant :

- son état (en veille, en cours de réception de données depuis le système d'information de l'usine de production, en cours d'envoi de données au système d'information de l'usine de production, réception de données depuis l'informatique d'un concessionnaire, envoi de données vers l'informatique d'un concessionnaire, transformation des données, etc.) ;
- des statistiques concernant son activité (nombre d'information envoyées, reçues, nombre d'erreurs, etc.).

Afin de superviser le middleware, l'équipe informatique d'exploitation doit pouvoir installer un petit logiciel sur chaque poste de travail. Ce petit logiciel doit pouvoir interroger le middleware via le réseau IP de l'entreprise, pour savoir si ce dernier fonctionne normalement (ou s'il est en erreur), pour savoir combien de connecteurs sont lancés, et pour connaître l'état ou avoir des statistiques d'activité d'un ou de tous les connecteurs.

Schéma simplifié de l'architecture du middleware



Couverture fonctionnelle du travail à réaliser par les auditeurs de l'UE NSY103 de Nancy et de Metz

Les auditeurs n'ont pas à réaliser 100% du middleware. Leur travail se focalise uniquement sur la programmation système de :

- l'infrastructure logicielle qui tourne sur le serveur du middleware,
- et du logiciel à installer sur les postes des informaticiens qui ont besoin de superviser le système.

Les auditeurs n'auront pas à traiter les problématiques suivantes :

- ils n'auront pas à s'occuper des programmes qui tournent sur le système d'information de l'usine de production, ni des programmes qui tournent chez les concessionnaires ;
- ils n'auront pas à s'occuper du travail d'adaptation de protocole (exemple : traduire un flux XML et générer un fichier où les informations sont distinguées à l'aide d'un caractère séparateur) ;
- ils n'ont pas à s'occuper des différents dialogues middleware<->succursales. Aussi, on supposera l'existence de fonctions "*dialogue_par_bal()*", "*dialogue_par_socket()*", "*dialogue_par_webservices()*", qui seront appelées par le code des connecteurs.

Barème

La moitié de la note sera évaluée sur la description de l'architecture du système cible. Dans son rapport, le candidat décrira quels mécanismes il met en oeuvre pour résoudre chaque problématique de communication, de multitâche, d'accès aux données partagées, etc.

L'autre moitié sera attribuée sur l'évaluation des algorithmes (qui permettent le la rédaction des programmes constituant le middleware et le logiciel client de surveillance) proposés par le candidat.

Bonus : si le candidat fournit du code en C opérationnel en complément ou à place des algorithmes, un "bonus" lui sera attribué. Ce bonus pourra rattraper une éventuelle moyenne tangente à 10 après l'examen.

Modalités de remise du projet

Le projet devra être impérativement remis sous forme électronique (sur CD-Rom ou par e-mail) AVANT l'examen final du NSY103. Dans le pire des cas, un CD-ROM pourra être remis à la personne qui surveille l'examen AVANT la distribution des sujets. Le candidat signera alors une feuille d'émargement. Sinon, l'ensemble du projet pourra être envoyé par email à l'adresse emmanuel@desvigne.org avant l'examen. Chaque réception d'e-mail sera suivi d'accusé de réception de la part du correcteur, indiquant que le projet a bien été reçu.

Le rapport du projet (obligatoire) pourra être rédigé dans un document en texte brut (ASCII ou UTF8), ou en RTF, ou au format MS-Wod doc 1997-2003, ou préférentiellement au format Oasis Open Document (format odt). Les algorihtmes pourront faire partie de ce document, ou être mis dans des fichiers texte séparés. Enfin, l'éventuel code source en C sera fourni dans des fichiers séparés ayant une extension ".c", ".h", etc.

L'ensemble de tous ces documents pourra être intégré dans un seul fichier (archive 7Z, ZIP, RAR, .tar.gz, ou .tar.bz2).