CMMAI : application de supervision à distance d'une ligne de conditionnement temps réel

Décembre 2010

Chapitre 1

Complément de spécifications

Ce chapitre décrit les éléments de spécifications qui ne sont pas indiqués dans le cahier des charges. Ce complément à pour objectif de préciser le comportement de l'application et a donc une influence directe sur la réalisation du projet.

Dans l'atelier, les lampes des trois couleurs (rouge, orange et verte) sont reliées à un contrôleur tricolore auquel on demande d'allumer l'une des lampes (il ne peut y avoir qu'une lampe allumée à la fois).

Le serveur sera hébergé sur le poste *VxWorks*.

1.1 Impression des étiquettes

- Les imprimantes sont sollicitées en tour à tour, ceci permet d'équilibrer la charge sur les imprimantes et de répartir l'usure.
- L'état de fonctionnement de l'imprimante est testé lors de l'impression d'une étiquette. Si l'impression s'est révélée impossible, alors la seconde imprimante sera sollicitée.

1.2 Journaux d'exploitation

L'enregistrement d'un message dans les journaux d'exploitation s'effectue suffisament rapidement pour considérer la date d'enregistrement du message comme la date d'un événement (imprecision inférieuse à une seconde, tandis que l'horloge est précise à la seconde).

1.3 Client windows

Le superviseur, qui accède à l'application à partir du logiciel client sous windows doit indiquer les paramètres de configuration suivants :

Au lancement de l'application :
- Le seuil de pièces deffectueuses acceptées

Pour chaque nouveau lot à traiter :
- Le code opérateur (permet d'identifier le superviseur),
- la référence du lot (permet d'identifier le lot traité),
- le nombre de cartons du lot,
- le nombre de pièces par carton.

1.4 Liste des erreurs et anomalies émises par le serveur

- Le seuil des pièces deffectueuses est atteint,
- La file de cartons est pleine,
- Il n'y a plus de cartons disponible,
- Il n'y a plus de pièce disponible,
- Une imprimante est en panne (*anomalie*).

1.5 Traitement des erreurs

Le processus de gestion des erreurs est le suivant :

1. Fermeture du clapet
2. Allumer le voyant rouge de l'atelier
3. Génération d'un message destiné au superviseur
4. Écriture du message dans les journaux d'exploitation
5. Passage en mode interactif (mode dialogue) côté client : demande de reprise d'activité ou terminaison de l'application.

1.6 Traitement des anomalies

1. Allumer le voyant orange de l'atelier
2. Génération d'un message destiné au superviseur
3. Écriture du message dans les journaux d'exploitation

1.7 Transactions sur le réseau

Le protocole retenu repose sur des messages de type texte, afin de s'abstraire des problèmatiques d'architecture materielle entre les différents systèmes : le serveur sous *VxWorks* (architecture *16 bits*) et le client sous Windows (architecture *intel 32 ou 64 bits*).

Liste des types de messages du protocole :

- Client vers serveur

  Configuration décrit la configuration du traitement d'un lot,

  Launch indique le lancement du traitement d'un lot,

  Stop demande la fin de l'application à la fin du traitement du lot courrant,

  Resume demande la reprise de l'application (après une erreur).
- Serveur vers client

  Log envoie un message du journal d'exploitation,

  Warnings envoie un code d'alerte décrivant une anomalie,

  Error envoie un code décrivant une erreur,

  Accepted envoie le nombre de pièces acceptées dans le lot,

  Rejected envoie le nombre de pièces rejetées dans le lot.

Chapitre 2

Notice d'intégration

Ce chapitre présente les différents éléments constituant l'application qui sont partagés entre les lots de travail, les conventions à respecter lors de la réalisation du projet afin de faciliter l'intégration et l'organisation de l'étape d'intégration.

2.1 Découpage en lots

- **Lot n◦1** : Communication (serveur sockets, journal d'événements) (Maxime, paul),
- **Lot n◦2** : Gestion du conditionnement et de l'impression (Yoann, Monica),
- **Lot n◦3** : Tâche mère, IHM (client windows), gestion de l'arrêt d'urgence (Martin, Etienne).

2.2 Ressources partagées

Nous avons pris la décision de donner la responsabilité des ressources partagées entre différentes tâches à la tâche mère $masterTask$.

Les ressources partagées sont les suivantes :

- *endSync* sémaphore de synchronisation notifiant la fin de l'application,
- $boxHandlingRequest$ sémaphore de synchronisation indiquant à la tache $boxManager$ qu'elle doit activer le traitement du conditionnement,
- *boxesQueue* file d'attente des cartons pleins prêt à être étiquetés par la tache $printManager$,
- *eventsQueue* file d'attente des événements survenus lors de l'exploitation,
- $logsEventQueue$ file d'attente des messages à écrire dans le journal des événements,
- *socket* socket réseau permettant la communication entre le client (opérateur manipulant l'application windows) et le serveur sur l'application en temps réel.

2.3 Types et données partagées

Les données, structures et type de données partagées sont regroupés dans un fichier d'en-tête générique $boxingServer.h$, en voici les principaux :

- les structures de données manipulées par les files d'attente ($event_msg_t$ et $boxesQueueMsg_t$),
- les structures des mémoires partagées ($boxData_t$ et $settings_t$),
- les événements survenant lors de l'exploitation sont représentés par une valeur constante, dont la valeur numérique peut être définie manuellement lorsque l'événement est transmis sur le réseau à destination du client Windows.

2.4 Guide de style

Nous avons mis en place des recommandations de développement pour nous assurer que le code produit par les trois équipe soit cohérent et que les conflits seront limités. Ainsi, un guide de style a été rédigé pour mettre l'accent sur la nomenclature des symbôles utilisés dans le code, l'encapsulation des opérations internes des tâches, et le respect de conventions facilitant les échanges entre les taches.

2.5 Interfaces

Nous avons également mis en place des interface permettant d'abstraire la manipulation des périphériques (capteurs, imprimantes, clapets) dans l'optique de simplifier les tests et la simulation de l'application. On supposera que pour passer l'application en production, il nous suffira d'utiliser une implémentation de ces interfaces contrôlant les périphériques, en utilisant, par exemple, des règles de compilation différentes dans le $Makefile$ de l'application.

2.6 Processus d'intégration

Le processus d'intégration a lui-même été découpé en plusieurs *lots* de manière à répartir le travail. La mise en place en amont du guide de style et des conventions à respecter simplifie grandement l'opération, qui devient essentiellement une tâche de câblage (réaliser les liaisons entre les différentes tâches) et de correction du code.

Ainsi, les opérations suivantes ont pu être traitées simultanément :

- Rédaction du plan de test d'intégration et élaboration des tests,
- câblage des modules,
- construction du simulateur,
- préparation du déploiement de l'application.

2.7 Déploiement

Le déploiement de l'application à nécessité la configuration de l'environnement de travail *Windriver Workbench* pour utiliser correctement le simulateur *VxWorks*.

Il faut installer une carte réseau virtuelle `WRTAP`, pour avoir la possibilité de communiquer entre l'hôte et la cible (à savoir la machine virtuelle VxWorks). On doit donner à la carte réseau virtuelle l'adresse `192.168.200.254`. Il faut ensuite installer le service `vxsimnetd`. Pour cela, il s'agit d'utiliser l'exécutable `vxsimnetds_inst`, qui se trouve dans le répertoire d'installation de Windriver Workbench.

Ensuite, pour configurer le simulateur, placer l'IDE Workbench dans la vue Remote Systems (en haut à droite de la fenêtre). Sélectionner le simulateur `vxsirn0`. Dans le menu contextuel qui est proposé lors d'un clic droit sur cette machine virtuelle, choisir l'option *propriété*. Dans le premier onglet, choisir `Advanced Boot Parameters`. Sélectionner `simnet` dans la liste déroulante `Boot device`. Fixer l'adresse IP de la machine virtuelle à `192.168.200.1`. Valider et lancer le simulateur, qui dispose alors de l'IP locale ainsi définie.

Il s'agit alors de tester la configuration mise en place. La commande `ping 192.168.200.1` doit répondre, sur une intallation de Windows standard, quatre lignes, avec des temps très faible, puisque le roundtrip mesuré est effectué sur la boucle locale.

Le client Windows a été réalisé à l'aide du Framework Qt pour C++. Nous avons donc choisi une stratégie de déploiement adaptée à des postes n'ayant pas les pré-requis (bibliothèques Qt)

disponibles.

2.7.1 Installation de la bibliothèque Qt

Qt est une bibliothèque multi-plateforme, disponible pour Windows, MacOS, GNU/Linux, proposant, entre autre, un ensemble de composants graphiques et réseaux. Notre application utilise cette bibliothèque, afin de simplifier et de rendre portable les parties réseau et interface graphique. Le guide ci-dessous permet d'installer, à partir d'une installation vierge de Windows (toute version), la bibliothèque Qt, son environnement de développement, et l'application permettant de contrôler le système à distance.

- Télécharger Qt pour Windows à l'adresse :
  `http://qt.nokia.com/downloads/sdk-windows-cpp`.
- Double-cliquer sur le fichier téléchargé, et suivre les étapes d'installation (pour une machine x86, sur un Windows XP, Vista ou Seven, cette procédure consiste à cliquer sur suivant à chaque écran, et à accepter la licence LGPL proposée).
- Si l'on désire compiler le programme à partir des sources, ouvrir le fichier d'extension `.pro` dans QtCreator, l'IDE de Qt, et cliquer sur la flèche verte en bas à droite de l'écran. Cela compile et lance le programme. Ceci est la méthode recommandée pour utiliser ou tester le programme de contrôle, puisqu'il installe de fait correctement l'environnement Qt.
- Si l'on désire exécuter le programme sans l'avoir compilé préalablement, il faut s'assurer qu'il a été compilé pour la bonne architecture, et pour le bon type de système. Il s'agit, comme un programme Windows classique, de double cliquer sur l'exécutable généré par la compilation. Il est impératif que les bibliothèques Qt soient installées dans le système, et soient dans le chemin de recherche des bibliothèques partagées. De nombreuses entreprises d'informatique embarquée utilisent Qt, ce point ne devrait donc pas trop poser de problème.