

Evaluation d'une messagerie industrielle pour le contrôle de périphériques son

Hans-Nikolas Locher

6 février 2003

Table des matières

1	Organisation du stage	4
1.1	Cadre du stage	4
1.2	Le propos du stage	4
2	Présentation du travail à accomplir	5
2.1	Présentation des logiciels de supervision	5
2.1.1	Principales caractéristiques	5
2.1.2	La norme TASE.2	5
2.1.3	Les autres offres du marché	7
2.1.4	Les travaux au CEDRIC sur le contrôle-commande	8
2.1.5	Le logiciel OpenTAZ	9
2.2	L'application multimédia	9
2.2.1	L'instrument virtuel	9
2.2.2	Présentation de jMax	10
3	Le travail de développement	11
3.1	Description technique de jMax	11
3.2	Description technique de OpenTAZ	12
3.3	Les solutions d'interfaçage	12
3.3.1	Un seul processus	12
3.3.2	Communication par IPC	12
3.3.3	Communication via CORBA	13
4	L'interface moritAz	13
4.1	La formule retenue	14
4.2	Exemple de fonctionnement	14
4.3	Limites du fonctionnement actuel	19
4.4	Perspectives pour une seconde version	19
4.4.1	Rappel des contraintes	19
4.4.2	Un peu plus de souplesse	20
4.5	Perspectives pour une version ultérieure	21
4.6	Conclusion	21
5	Travail de mesures, d'installation, de configuration	22
5.1	Une séance de mesure	22
5.1.1	Introduction	22
5.1.2	La démarche	22
5.1.3	Commentaire sur les mesures	23
5.2	Pilotage des objets RTP : collaboration avec Nicolas Bouillot	23
5.3	L'informatique au quotidien	26
5.3.1	Déménager	26
5.3.2	Ne pas être root	27

6	Travail de réflexion	27
6.1	Lectures des articles	27
6.2	Travail de compréhension	28
6.3	Adéquation des sémantiques	29
6.3.1	Liste des variables	29
6.3.2	Estampillage	30
6.4	Interactions	31
6.5	Modèle objet	31
6.6	L'avenir	32
7	Travail de présentation	32
7.1	confection du rapport et rédaction intermédiaire	32
7.1.1	Le choix des outils d'édition	32
7.2	Préparation à la soutenance du stage	33
7.3	Démonstration avec trois machines devant François Dechelle	33
8	Conclusion : Les acquis du stage	35
8.1	Ma pratique du développement	35
8.2	Faire simple	35
8.3	L'expérience en laboratoire	36
8.4	Quelques contraintes d'organisation	36
8.4.1	Une ressource unique : la clef	36
9	Remerciements	36

1 Organisation du stage

1.1 Cadre du stage

5 Ce stage a pour objectif la validation du DUT d'informatique (option Génie Informatique) au CNAM. Il compense l'expérience professionnelle dans le domaine.

6 Ma pratique d'opérateur projectionniste au Pathé Wepler, me permet d'appréhender la réalité du monde du travail.

7 Ce stage, réalisé en interne, à fait l'objet de l'agrément pédagogique de Madame Joëlle Delacroix-Gouin, responsable du Cycle A en Informatique au CNAM.

8 Éric Gressier-Soudan, membre du laboratoire CEDRIC (centre de recherche en informatique du CNAM) et professeur en informatique au CNAM, en est le tuteur.

10 La durée prévue du stage est de trois mois. Pour me permettre de continuer à travailler, le stage a été aménagé sur une période de six mois à mi-temps. Il se déroule du 15 juillet 2002 au 15 janvier 2003 au laboratoire CEDRIC, 55, rue de Turbigo, 75003 PARIS. Je dispose d'un poste et d'une machine au bureau des stagiaires au 5ème étage. J'ai organisé mon planning en venant le jeudi et le vendredi, de 9h à 19h.

1.2 Le propos du stage

13 Ce stage fait suite à de nombreux travaux au CEDRIC sur les protocoles industriels de contrôle-commande, articulés essentiellement sur le protocole ISO/MMS.

14 TASE.2 est une norme d'accompagnement de MMS, et normalise la commande de machine/outils. Un prototype de messagerie, OPENTAZ conforme à la projection en IDL/Corba de la norme TASE.2 a été écrit en C++ par Erwan Becquet.

15 Il s'agit d'étudier la faisabilité et la pertinence de son utilisation, pour le contrôle-commande d'instruments virtuels dans le cadre d'un concert virtuel réparti sur l'Internet.

16 La plate-forme sonore est le logiciel JMAX, développé par l'IRCAM, un langage de programmation graphique. Ce logiciel est déjà utilisé dans le cadre d'autres travaux sur le concert, notamment pour l'échange de flux audio avec le protocole RTP par Nicolas Bouillot (FIXME référence).

17 Le travail de programmation consiste à trouver un interfaçage convenable entre JMAX et OPENTAZ.

18 Le reste de la démarche consiste à analyser les avantages et les inconvénients de la sémantique d'échange apportée par OPENTAZ pour le contrôle d'objets produisant du son.

2 Présentation du travail à accomplir

2.1 Présentation des logiciels de supervision

2.1.1 Principales caractéristiques

24 Le but de ce genre de logiciel, est de permettre d'avoir une vue d'ensemble, et un moyen de contrôle sur des équipements ou des applications réparties dans l'espace. Les termes que l'on rencontre sont "timed messaging service", ou "real-time data exchange".

25 On peut relier ce genre d'installation à une base de donnée. L'acronyme "SCADA" signifie Supervisory Control And Data Acquisition (Contrôle de supervision et acquisition de données). Il résume assez bien l'ensemble des préoccupations au centre de ce type de logiciel.

26 Les besoins peuvent être assez différents selon les applications. Celles-ci peuvent aller du contrôle de parc informatique (éventuellement hétérogène) au contrôle d'automates, en passant par le contrôle distant de paramètres sur un objet multimédia!!!

27 Le protocole TASE.2 et sa transcription en CORBA/C++ a pour objectif premier une aide à la supervision dans une centrale électrique. Les contraintes temps réel sont lâches (FIXME Combien) de l'ordre de la milliseconde. SNMP est destiné à l'administration d'équipements réseau.

2.1.2 La norme TASE.2

29 TASE.2 est la version de l'IEC de la norme ICCP¹. Il s'agit d'un "standard compagnon" de MMS, tout deux définis par l'ISO. Son application typique est celle d'aide à l'automatisation d'une centrale électrique petite ou moyenne.

30 MMS est un protocole d'échange industriel en mode client/serveur basé sur l'idée de périphérique virtuel (Virtual Manufacturing Device). Le VMD peut être un automate, par exemple, ou une machine à commande numérique.

31 TASE.2 propose un modèle d'échange client-serveur basé sur des contrats. Le serveur est un VCC (Virtual Control Center) et fournit les données ou les services, tandis que le client y fait appel. Une application peut jouer les rôles de l'un ou de l'autre, ou encore des deux. Le VCC peut agir comme interface avec un équipement.

32 Le client contacte le serveur et définit avec lui une ou plusieurs associations (par exemple avec des qualités de services différentes). La communication avec le serveur se fera via cette association. Le protocole fonctionne donc en mode connecté.

33 Le serveur possède un certain nombre de `DataValue`, et une liste de clients

¹La norme TASE.1 était basée sur la norme ELCOM 90

connus, ainsi qu'une table de correspondance établissant les droits en lecture et/ou écriture du client sur chaque donnée.

34 Un client peut être une console de supervision (accès en lecture des données) ou de contrôle (accès en écriture).

35 Pour prendre un exemple parlant et fréquemment donné, le serveur peut être lié à un bras articulé, et les données mises à disposition peuvent être les angles des articulations. Un poste de monitoring peut récupérer la valeur des angles lorsqu'ils changent, et éventuellement donner une représentation sous la forme d'une liste de données, ou pourquoi pas une modélisation en trois dimensions. Un poste de supervision peut proposer de modifier les angles, avec l'interface appropriée.

36 Le client peut demander une liste des données que le serveur met à sa disposition. Une `DataValue` peut contenir un `IndicationPoint`, un `ControlPoint` ou un `ProtectionEvent`. Un `IndicationPoint` peut contenir un état, un nombre entier ou un flottant (`state`, `discrete` ou `float`).

37 Du côté du client, on peut regrouper un ensemble de données dans un `DataSet`.

38 TASE.2 propose plusieurs sémantiques d'échange de données. On peut récupérer la valeur d'une donnée à la suite d'un `get`, comme un simple appel procédural. On peut récupérer un ensemble de valeurs périodiquement (periodic report), à chaque fois qu'elle change (report by exception) ou en fonction d'une condition sur sa valeur. Pour ce faire, on définit un `DataSetTransferSet` muni des paramètres voulus. En particulier, on peut demander que des changements successifs soient agglomérés.

39 On peut également définir un `TimeSerieTransferSet` qui permet d'obtenir les différentes valeurs d'une `DataValue` dans un intervalle de temps.

40 Les données sont accompagnées d'informations de fraîcheur, de fiabilité (valeur calculée ou mesurée), nombre de changements de valeurs...

41 La norme définit la projection des différents éléments sur des types MMS, et laisse entrevoir une API, sans la décrire formellement.

42 La norme se découpe en neuf blocs de conformité

- . bloc 1 : ensemble minimal de fonctions avec les services de gestion de données et échanges périodiques
- . bloc 2 : ajoute au bloc 1 la sémantique d'exception (report by exception)
- . bloc 3 : permet une stratégie d'agrégation et compactage des données échangées
- . bloc 4 : échanges de messages binaires ou en texte libre
- . bloc 5 : contrôle de périphériques

- . bloc 6 : contrôle de programmes
- . bloc 7 : gestion des évènements
- . bloc 8 : "Comptabilité" des échanges
- . bloc 9 : ajoute l'abstraction de TimeSerie, les différentes valeur d'une même variable dans le temps.

2.1.3 Les autres offres du marché

SNMP 46 SNMP est un protocole défini par la RFC 1157, et se propose d'être un protocole standard de management de réseaux. Il permet de stocker des données dans une base de données : la MIB. Le format des enregistrements dans la MIB est défini en ASN.1.

47 Les ressources référencées dans la MIB possèdent un nom unique, défini dans une organisation arborescente normalisée.

48 On distingue les *agents* et le *manager*. Ils implémentent tous les deux le protocole SNMP. Les agents sont les entités qui interfacent le matériel à contrôler. Il peut s'agir de routeurs, de ponts, etc, mais également d'une rampe de diode branchée sur le port parallèle d'un PC. (FIXME lien vers linuxmag).

49 Les données à manager dépendent des besoins. Il s'agit en général de statistiques, par exemple le nombre de paquets échangés. Dans le cas de ponts formant un arbre recouvrant, une donnée peut indiquer si le pont est actif, inhibé ou en défaut.

Le framework JMX 53 Il est présenté par SUN MicrosystemsTM et propose une architecture de management assez souple. Il est basé sur une approche composant, celle des Enterprise JavaBeans.

54 JMX est défini sur trois niveaux.

55 Dans le niveau instrumentation on développe des objets ou des ensembles d'objets qui encapsulent la ressource à manager et forment un **ManagedBean**. Les principes à respecter sont propres à Java et diffèrent légèrement selon que l'on veut que l'interface soit statique ou dynamique.

56 Le niveau agent propose la notion de serveur de **MBean**, les composants définis à l'étape précédente. Un agent s'exécute typiquement dans une JVM. C'est le code de l'application proprement dite.

57 Le niveau service propose deux abstractions, celle d'adaptateur de protocole (communication via HTTP en formatant l'état du **MBean** en HTML, ou traduction vers des messages SNMP par exemple), et celle de connecteur (adaptation vers RMI ou CORBA).

2.1.4 Les travaux au CEDRIC sur le contrôle-commande

61 FIXME (insérer citation HDR EGS)

Présentation de CORBA 64 Les nombreuses couches présentes dans MMS ont été remplacées par CORBA.

65 CORBA est une norme de l'OMG, un consortium d'acteurs de l'industrie du logiciel. Il s'agit d'un middleware qui fournit une sémantique d'invocation de méthodes à distance multiplateforme et multilingage.

66 Pour être schématique, il fonctionne sur deux idées essentielles. Le mécanisme de procuration, ou d'objet mandataire, et d'un langage homogène de description d'interface.

67 Chaque objet distant est représenté localement par une souche, qui dispose de la même interface. Pour invoquer une méthode distante, on invoque en fait la méthode homonyme sur l'objet subrogé, qui lui s'acquiesce du mécanisme à mettre en place pour contacter l'objet distant, que ce soit le marshalling des données ou les appels à la couche réseau.

68 Le marshalling, ou sérialisation, utilise la norme CDR (Common Data Representation) de représentation des données.

69 Le langage de définition d'interface s'intitule IDL comme (Interface Description Language). Sa syntaxe ressemble au C++. La notion de paramètre *in*, *out* ou *inout* ressemble à celle du langage Ada. La norme propose une abstraction de liste intitulée *sequence*.

70 La norme définit également une projection pour un certain nombre de langages. Les abstractions proposées donnent lieu à des constructions plus ou moins heureuses selon les possibilités du langage cible.

Projection de MMS sur CORBA 72 (FIXME ref articles)(FIXME HDR) MMS est défini au dessus de MAP. La propagation de TCP/IP dans le monde industriel donne une direction que plusieurs moyens permettent d'atteindre. On peut réimplanter directement MMS au dessus de TCP/IP, implanter l'ensemble des couches hautes OSI au dessus de TCP/IP avec une interface définie par la RFC 1006, placer MMS au dessus des RPC ou encore MMS au dessus de CORBA.

73 (FIXME schéma issus de la HDR)

74 Le travail sur MMS et Corba est basé sur une réflexion sur l'approche composant. Le fait d'exporter une interface permet d'utiliser la commande numérique d'une machine comme un *plug-in*, c'est à dire pouvant être chargé et déchargé sans recompilation. Le VMD est un objet fournissant des services utilisables par un client (user). On reprend les concepts en utilisant un bus

logiciel, en l'occurrence CORBA qui remplace les nombreuses couches protocolaires.

75 Pour utiliser Corba, on déclare les interfaces dans des fichiers IDL, qui sont précompilés pour générer les souches nécessaire dans la langage cible. Les projections les plus courantes se font vers les langages C++ et Java.

2.1.5 Le logiciel OpenTAZ

Le travail en CORBA/C++ 78 OPENTAZ à été programmé en C++, en utilisant l'ORB Mico, qui est conforme à la spécification CORBA 2.3.

79 L'interface graphique du client OPENTAZ a été mis en place grâce au toolkit graphique FLTK.

Les limites de l'implantation 82 OPENTAZ ne couvre pas l'ensemble de la norme TASE.2. Il est conforme aux deux premiers blocs avec quelques limitations.

83 En particulier, seul le type `IndicationPoint` est implanté, le `ControlPoint` étant lié au bloc 5 (contrôle de périphériques).

84 Le constructeur pour créer un `IndicationPoint` de type `float` n'est pas codé, bien que des indications soient données dans les commentaires pour le faire.

86 FIXME : insertion image du client ?

Un client graphique 89 Le client graphique permet de bien comprendre l'interaction avec le serveur. Il se présente sous la forme d'une arborescence sur la gauche, et d'une fenêtre avec les propriétés de l'objet actif à droite.

90 On peut ajouter un serveur, créer une association, récupérer la liste des `DataValue` disponibles, créer des `DataSet`, des `DataSetTransferSet`, de les mettre en route, de préciser les options voulues dans les reports.

91 Il ne permet pas de visualiser l'arrivée des reports.

92 La bibliothèque a un peu évoluée, ce qui m'a valu quelques difficultés de compilation.

2.2 L'application multimédia

2.2.1 L'instrument virtuel

96 FIXME : insertion du texte écrit dimanche!!!

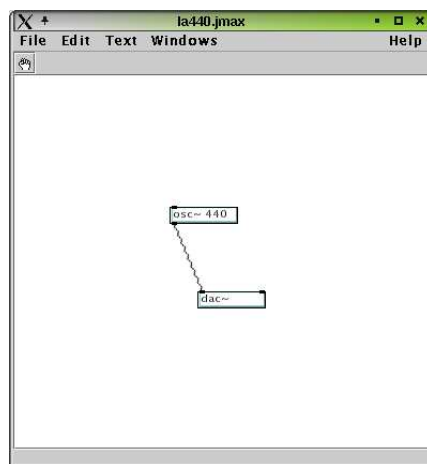


FIG. 1 – *Un diapason en activité*

2.2.2 Présentation de jMax

99 L'IRCAM définit JMAX comme un environnement de programmation visuelle pour des applications musicales et multimédia temps réel.

100 Il permet de faire une grande variété d'activités liées au son. On dispose d'objets de différentes natures. On peut le voir comme un établi avec une boîte à outils.

101 L'unité de travail est le *patch*. Il s'agit d'une sorte de canevas sur lequel on peut poser des objets, et les relier entre eux à l'aide de la souris. Selon la nature de l'objet, il peut disposer d'un nombre variables d'*inlet*, sur le bord supérieur, qui permettent d'adresser un message à l'objet, et d'*outlet* sur le bord inférieur, qui permettent à l'objet d'envoyer des messages aux objets qui lui sont reliés.

102 Un objet de contrôle peut afficher un message comme un nombre, ou une chaîne de caractère, tandis qu'un objet DSP a une sémantique de flux. Le travail de programmation pour coder un objet est assez différent dans le second cas, puisqu'il faut écrire un traitement élémentaire qui sera appelé périodiquement par FTS.

103 Il y a plusieurs catégories d'objets. Les plus remarquables visuellement sont les objets GUI, comme le slider, qui émule un potentiomètre, ou le bouton poussoir, qui envoie le message "bang" sur son *outlet*. "bang" est le message de déclenchement.

104 Mais on peut également trouver des objets plus proches des langages de programmation. Il s'agit de rectangles, dans lesquels on écrit le nom de l'objet et ses paramètres s'il en a.

108 Sur la figure 1, on peut voir un oscillateur à 440 Hz relié à un convertisseur numérique-analogique, c'est à dire la sortie audio de la carte-son. Ces

sont tous les deux des objets DSP, ils échangent des flux. On peut remarquer que le trait les reliant est ondulé, ce qui indique que l'échange des flux est actif.

3 Le travail de développement

3.1 Description technique de jMax

111 JMAX est issu du logiciel Max, conçu par W. Puckette. A l'époque, aucun compilateur C++ n'était disponible sur la plate-forme employée, et le logiciel a été écrit en C. JMAX émule la programmation orienté objet à l'aide de callbacks. JMAX a été écrit par François Dechelle, Norbert Schell, Nicolas Orio et Riccardo Borghesi.

112 L'interface graphique JMAX est programmée en Java et communique via une socket avec le back-end FTS (pour Faster Than Sound).

113 La configuration de la compilation se fait en appelant `MAKE` avec des paramètres, par exemple

```
make ARCH=i686-Linux
```

pour une plate-forme Intel/GNU/Linux.

114 Des morceaux de `Makefile` adapté à chaque plate-forme sont incluse dans le `Makefile` principal en fonction de l'architecture.

115 Le plus simple pour commencer un objet existant est de copier un package existant et de le modifier, et de l'ajouter dans le `Makefile` du repertoire contenant les packages.

116 Un package comprend en général plusieurs objets. Il y a donc un fichier `Source` qui liste les fichiers à compiler.

117 Une documentation existe pour donner les appels aux principales fonctions de FTS.

118 Il faut écrire le contenu des objets en s'inspirant des exemples en général avec les objets suivants :

119

- . Un fichier `Projet`

- . Un ou plusieurs fichiers par objet, selon la modularité voulue

120 L'exécution est paramétrée par un fichier de configuration `.jmaxrc` dans le repertoire de base de l'utilisateur.

3.2 Description technique de OpenTAZ

122 L'interface de programmation sous-jacente à la norme TASE.2 a été transcrite en une spécification IDL (en réalité, la norme précise les échanges FIXME référence à la description de OpenTAZ).

123 Erwan Becquet a réalisé une implémentation en C++ avec CORBA de cette spécification. Le travail a été effectué sous GNU/Linux avec l'ORB Mico 2.3.5. J'utilise la bibliothèque de Mico 2.3.7 sans difficulté, si ce n'est la correction des makefiles.

124 Le système de compilation est basé sur les outils de développement GNU : AUTOCONF et AUTOMAKE. Les fichiers de configuration sont générés par l'outil Gnome de programmation graphique ANJUTA. Les bibliothèques n'étaient cherchées que dans les emplacements standards, d'où quelques bricolages à la main.

3.3 Les solutions d'interfaçage

3.3.1 Un seul processus

127 Les systèmes de compilation sont divergents, et pour un novice dans l'utilisation des AUTOTOOLS, difficiles à faire converger.

128 Une autre difficulté était celle de l'interfaçage du C et du C++. Ce problème se résout en forçant le mangling "C" par une directive de compilation

```
#extern "C"
```

judicieusement placé dans les sources.

129 Un autre problème est le coût assez élevé du lancement de la messagerie, et la difficulté pour lancer les différentes parties dans le bon ordre (Le service de nom, le serveur, le client).

130 Une dernière difficulté est que OPENTAZ définit des objets globaux, ce qui fait que le code n'est pas ré-entrant, ce qui interdit de placer plusieurs instances de messagerie dans le même espace d'adressage et donc dans le même processus.

131 Or, un objet JMAX est implémenté sous la forme d'une bibliothèque partagée, chargée par JMAX en fonction des indications dans le fichier de configuration du package.

3.3.2 Communication par IPC

133 La communication par socket ou par mémoire partagée résout les problèmes précédents en apportant son lot d'inconvénients.

134 Premièrement, la communication complique le code en ajoutant des couches d'abstraction. Elle affecte les performances, d'autant qu'il y a déjà



FIG. 2 – *Un jeu de mot discutable (MAX et MORITaZ)*

une communication par socket entre FTS et JMAX.

135 En revanche, si OPENTAZ s'exécute dans un processus séparé, les systèmes de compilation distincts ne sont plus un problème, et la retouche du code de la messagerie n'est pas nécessaire autrement que pour insérer un module de communication.

136 Les deux logiciels reposent sur une sémantique de messages, que ce soit les `get` ou `set` de TASE.2 ou les envois de messages sur les `inlets` et `outlets` de JMAX.

3.3.3 Communication via CORBA

138 La précédente solution est un compromis, qui permet également d'éviter de passer à nouveau par le bus logiciel pour dialoguer avec

OpenTAZ

139 Cette dernière solution a été rejetée très rapidement, le VCC devant être l'interface du périphérique son, et un second passage par l'ORB étant jugé pénalisant pour les performances.

140 A l'heure où je termine ce rapport, en échange de courrier électronique avec Erwan Becquet m'apprend qu'il travaille actuellement sur une encapsulation de son application en un composant conforme à CCM (Corba Component Model).

141 Le point à élucider est celui de la performance. Si l'ORB (ici MICO) sait éviter d'envoyer les appels locaux jusqu'à la couche réseau, cette solution mérite d'être reconsidérée.

4 L'interface moritAz

145 *Max et Moritz* est un conte moral versifié allemand, narrant les méfaits de deux garnements (on peut les apercevoir sur la figure 2). Le fait de devoir faire travailler ensemble JMAX et OPENTAZ, ou le mot important est TAZ, qui est homophonique à TASE, m'a donné envie d'appeler l'objet `moritAz`².

²Conversation privée avec Pierre-Yves Locher

4.1 La formule retenue

147 C'est la solution utilisant les sockets qui a finalement été retenue. Afin de limiter la baisse de performance, c'est les sockets locales (Unix) qui sont utilisées.

148 Dans cette première version, pour simplifier le codage, il a été décidé que chaque objet aurait exactement 4 inlets/outlets, et qu'il y en aurait 5 définis statiquement dans le serveur.

149 La facilité de maintenance du code a pesé pour beaucoup. Le rapprochement des systèmes de compilation étant aussi ardu que l'interfaçage lui-même.

150 Pour JMAX, un package MORITAZ est défini. Il ne met à disposition qu'une seule sorte d'objet pour le moment.

151 Un objet JMAX a été implémenté, en utilisant l'API de fts, `mtaz_rc`, ou `mtaz` vaut pour le nom du package et `rc` pour "remote control".

152 Il faut créer une fonction pour l'initialisation de la "classe", `mtaz_rc_instantiate`, une pour l'initialisation de chaque instance `mtaz_rc_init`. Dans cette fonction, on crée une socket qui se branche à celle du serveur, et un thread qui se met en écoute sur cette socket.

153 Du côté serveur, on attribue les cinq jeux de variables prédéfinies dans l'ordre d'arrivée des clients.

155 La classe `Data` disposait d'un constructeur qui permettait d'associer des fonctions différentes aux lectures et écritures en les inscrivant comme callbacks, mais sans moyen de lui communiquer des données. J'ai donc créé une classe `UserData` qui hérite de celle-ci, et qui met en place un mécanisme de callbacks similaire, en lui adjoignant un paramètre de plus pour des données utilisateur. Cela permet de mémoriser l'origine de la valeur, en lui associant le numéro du client et celui de l'inlet dont elle provient. Le callback que j'inscris assure la mise à jour du client OPENTAZ.

156 Cet héritage m'a permis également de redéfinir la méthode `write`. Celle-ci appelle d'abord le callback inscrit, puis appelle la méthode de la classe mère, assurant ainsi que les `Report` liés au changement de valeur sont déclenchés.

4.2 Exemple de fonctionnement

159 J'ai tout d'abord ajouté le package dans le fichier `.jmaxrc`.

161 TODO :insertion du code de `.jmaxrc`

166 Sur la figure 3, on peut voir qu'au démarrage de JMAX, le package est chargé.

167 Actuellement, il faut lancer tout d'abord la messagerie. Un script

```
opentaz
```

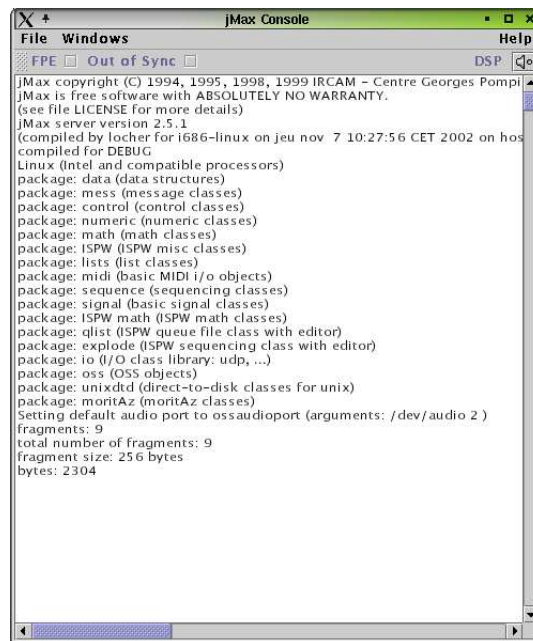


FIG. 3 – *Chargement du package moritAz*

écrit en bash accompagne les sources d'OPENTAZ, et a été modifié par mes soins pour une meilleure adéquation au problème. Ce script lance le service de nom CORBA, le serveur OPENTAZ puis le client graphique OPENTAZ. 168 Pour le moment, 5 jeux de 4 variables discrètes sont définis statiquement. Le client (voir la figure 4) les découvre après l'appel de `getDataNames`. On peut donc les voir apparaître avant même qu'ils ne soient liées à un objet dans JMAX.

170 Si JMAX n'a pas été lancé il faut le faire, mais il ne faut pas qu'un objet `moritAz` ai été fabriqué. Il faut ensuite créer ou ouvrir un patch avec un objet `mtaz_rc` pour `moritAz` remote control (voir la figure 5).

172 Si on modifie l'entrée de l'inlet 2 (figure 6), la valeur est mise à jour dans la messagerie.

175 Dans le client de la messagerie, on peut appuyer sur le bouton `GET` lorsque la valeur correspondante est sélectionnée. On vérifie que la mise à jour a été effectuée (figure 7).

178 Toujours dans le client, on peut mettre à jour la valeur. On inscrit 78 dans le champ prévu à cet effet, et on clique sur le bouton `SET` (figure 8).

180 On peut voir dans le patch JMAX que la sortie a été mise à jour, la valeur dans la boîte d'affichage liée à l'outlet 2 étant maintenant 78 (figure 9).

182 Lorsque l'on créé un `DataSet`, et un `DataSetTransferSet`, que l'on demande un `Report` par changement de valeur, on peut voir l'arrivée des

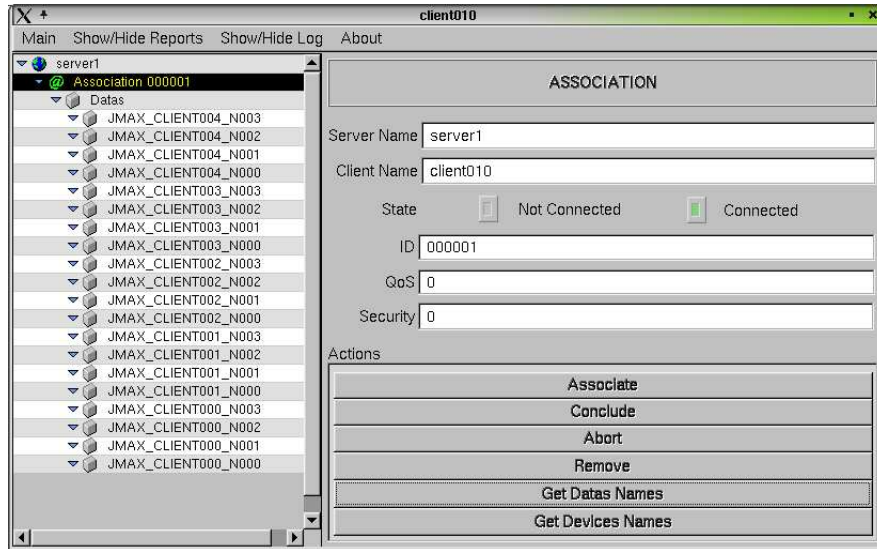


FIG. 4 – La fenêtre du client OpenTAZ avec les 5 jeux de 4 variables

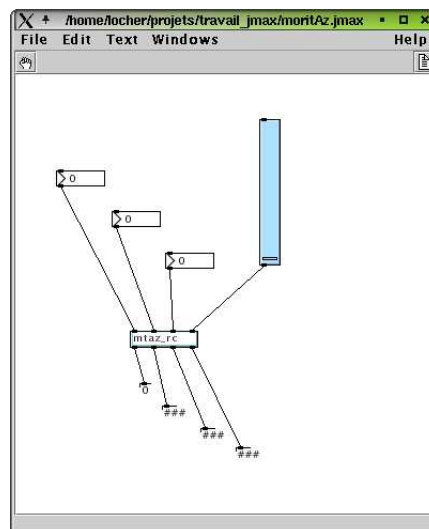


FIG. 5 – Un patch jMax avec l'objet moritAz

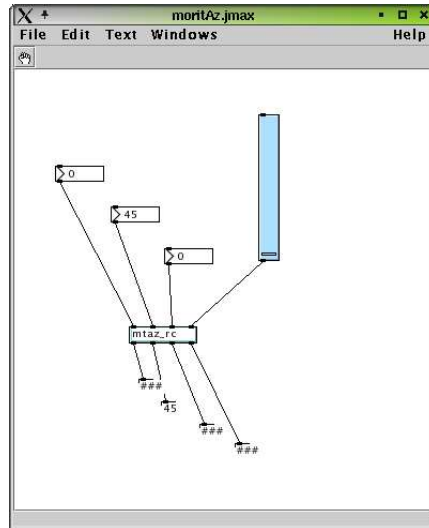


FIG. 6 – On entre 45 dans la boîte de saisie liée à l'inlet 2

The screenshot shows a software window titled 'client010' with a menu bar (Main, Show/Hide Reports, Show/Hide Log, About). On the left is a tree view under 'server1' containing 'Association 000001' and a 'Datas' folder with many sub-items. The 'JMAX_CLIENT000_N001' item is selected. On the right is a 'DATA' panel with the following fields:

- Name: JMAX_CLIENT000_N001
- Type: INDICATION_POINT Kind: DISCRETE
- Value: 45
- Extra Infos:
 - COV: N/A
 - TimeStamp: N/A
 - Validity: N/A Normal Value: N/A
 - Quality:
 - Current Source: N/A
 - Normal Source: N/A
- Actions:
 - Get
 - Set
 - Get Type

FIG. 7 – On consulte la valeur de la DataValue correspondante

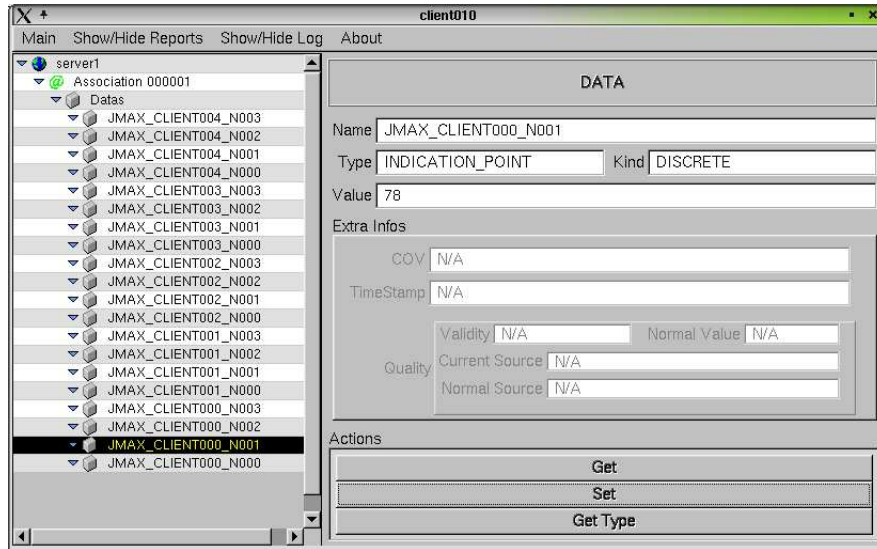


FIG. 8 – On appelle set avec 78

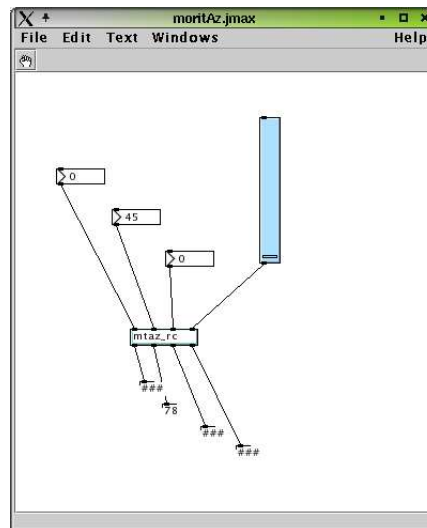


FIG. 9 – La sortie de la seconde inlet est à 78

reports dans les logs de l'application, mais rien n'est prévu pour montrer que la valeur à été mise à jour.

4.3 Limites du fonctionnement actuel

185 Dans le fonctionnement actuel, la messagerie définit 5 jeux de 4 paramètres. Lorsque l'on crée un objet `mtaz_rc`, il se connecte au serveur de messagerie via des socket, et est mappé sur le premier jeu disponible.

186 Ce fonctionnement est critiquable de plusieurs manières.

187 Tout d'abord, on ne peut pas faire beaucoup mieux que de numéroter les instances, ce qui dans le client, n'est pas très parlant. Ensuite, il est difficile de deviner quelle est l'instance représentée par un numéro, sauf si l'on sait dans quel ordre elles ont été créées.

188 Il pourrait être agréable d'instancier l'objet `moritAz` de la façon suivante :

```
mtaz_rc "@ns" "monNom" "param1" "param2"
```

189 L'adresse du service de nom est donné par "@ns". On aurait alors un objet avec deux inlets/outlets, qui s'appellerait "monNom", et qui mettrait à disposition du client de messagerie un paramètre "param1" et "param2".

4.4 Perspectives pour une seconde version

194 Le travail sur ce premier prototype minimal permet de faire surgir un certain nombre de points à améliorer.

195 Certaines proposition d'amélioration convergent d'ailleurs vers les même solutions.

196 Le choix de faire collaborer un objet `JMAX` avec le serveur `OPENTAZ` dans un autre processus via des sockets garde sa justification en première approche (voir `FIXME` référence).

197 Le fait de placer les deux parties du code dans le même espace d'adresse simplifie un certain nombre de points, éclairés plus loin.

4.4.1 Rappel des contraintes

199 L'amélioration du couplage aboutit à une amélioration de performances en limitant le nombre de thread exécutés en parallèle. En particulier les thread écoutant sur la socket disparaissent des deux cotés. On limite ainsi le nombre de commutations de contexte.

200 L'objet `moritAz` est adjoint à l'environnement de programmation visuelle comme un *plug-in*. Avec le système Unix GNU/Linux, il prend la forme

d'une bibliothèque partagée chargée dynamiquement. Le code doit donc être réentrant, en particulier celui d'OPENTAZ.

4.4.2 Un peu plus de souplesse

202 Par ailleurs, la bibliothèque Mico (c'est à dire l'ORB) doit être correctement initialisée. Cette initialisation se fait en général dans le `main` de l'application utilisatrice, en transmettant les arguments de la ligne de commande à une fonction conforme à la norme. Ici l'objet `jMax` n'est qu'un sous-ensemble de l'application et il faut procéder autrement.

203 Le point le plus sensible est la définition statique des variables d'un VCC selon la norme TASE.2. Le besoin d'un peu plus de souplesse se fait sentir dans l'utilisation de d'un objet dans un patch. Le fait de pouvoir créer et supprimer des serveurs VCC apporte un peu de dynamisme à une application configurée statiquement par construction.

204 On voit donc apparaître la notion d'objet VCC.

Travail sur OpenTAZ 206 Dans la version actuelle d'OPENTAZ, la granularité d'un processus est celle de VCC. Mon idée est que le processus devrait avoir une granularité plus large, en l'occurrence celle de *fabrique* de VCC.

207 Le code de OPENTAZ utilise la notion de singleton, d'où des variables statiques pour rendre les objets uniques. L'ennui est que la définition statique dans C++ se fait au niveau du module dans le processus.

208 Les variables contenant la référence de l'ORB et du service de nom pourraient devenir des variables d'instance de la `VCC_Factory`, tandis que la référence de la base de donnée devrait devenir une variable d'instance du VCC.

209 On pourrait ensuite créer un VCC en lui communiquant la liste des variables et de leurs types.³

Travail sur jMax (code de moritAz) 212 Si l'on souhaite que le VCC ne soit pas détruit et reconstruit trop fréquemment, il semble bon de définir deux types d'objets. Un objet contrôle distant, auquel on donne un nom, et un objet paramètre, qui permet de poser un point de contrôle n'importe où dans le patch.

213 On aurait donc une initialisation sous la forme suivante :

³On peut également supposer que certaines listes de variables soient prédéfinies. On pourrait par exemple définir un type `InstrumentVirtuel` comprenant le volume, le diapason, etc.

```
mtaz_rv nomVCC @ns nřPort etc ... liste des variables
```

```
mtaz_rv nomVCC @ns nřPort etc ... typePrédéfini
```

```
mtaz_param nomVCC nomParam
```

214 L'utilisateur peut alors créer des objets paramètre et les détruire sans relancer le VCC à chaque fois (ce qui évite les déconnexions du client OPENTAZ).

4.5 Perspectives pour une version ultérieure

217 Une façon plus complète de modulariser le code d'OPENTAZ serait de placer son code sous la forme d'une bibliothèque partagée. Une fois que le code est réentrant, une bonne partie du travail est fait. Il faut cependant définir une interface bien pensée, ce qui exige un travail important de spécification.

218 Il serait souhaitable dans la partie client de désolidariser le code d'affichage graphique, et le mécanisme du client lui-même. Je connais trop mal le client pour m'aventurer plus avant dans les hypothèses. Une approche type MVC paraît indiquée.

219 Une fois ce découplage effectué, pourquoi pas une mise en correspondance avec une interface plus appropriée au problème, des sliders, ou un objet jMax... ou encore une vraie console de mixage pilotée par midi ou par l'interface numérique ADAT.

220 Le fait d'avoir un objet client et un objet serveur disponibles dans la bibliothèque rendrait aisé le fait de créer des applicatifs ayant les deux interfaces.

4.6 Conclusion

224 La modularisation sous forme de bibliothèque permet d'éclaircir le fonctionnement du code.

225 Actuellement, on peut schématiser la répartition du code comme ceci.

226 TODO : schéma objets avec communication

227 TODO : schéma avec intégration du code.

5 Travail de mesures, d'installation, de configuration

5.1 Une séance de mesure

233 Un objectif intermédiaire fixé pendant le stage, à été de mesurer les performances des sur un réseau local de la messagerie. Pour cela, un compte à été créé sur une autre machine. Les fichiers n'étaient pas partagés à l'époque, et il fallait gérer la cohérence des différentes copies.

5.1.1 Introduction

235 Le but de ces mesures est d'avoir une quantification des performances
236 de la messagerie industrielle dans un contexte réel.

237 La version de la messagerie employée est la version 0.2 récupérée sur savanah en utilisant cvs.

239 Les essais en local ont été faits sur la machine *Oceanonix*, le 10 octobre 2002. Les mesures distantes ont été effectuée le 11 octobre 2002, le client étant lancé sur *oceanonix*, et le service de nom et le serveur sur *petitfour*.

241 La machine *oceanonix* est un PC équipé d'un pentium II 448MHz avec 160 Mo de mémoire vive. Le système d'exploitation est GNU/linux 2.4, distribution Suse 7.3. La machine *petitfour* est un PC équipé d'un pentium II 448MHz avec 128Mo de mémoire. La distribution de linux employée est la même.

243 Les machines sont reliées à deux switchs différents d'un réseau à 100Mbps, les switchs étant probablement reliés eux mêmes à un autre switch à l'échelle du local du 5ème étage. Le réseau était peu chargé au moment des mesures, peu d'utilisateurs étant présents.

5.1.2 La démarche

248 J'ai d'abord commencé à faire les mesures localement. Dans le code de la messagerie, il fallait inhiber le démarrage de l'interface graphique, et modifier certaines constantes pour valider les affichages temporels.

250 Joël Berthelin m'a ouvert un compte sur une quatrième machine, petitfour. Les comptes sur *ppc1* et *mais* n'étaient pas utilisables puisqu'il reste très peu de place sur le disque dans le premier cas, et que l'utilisateur travaille sous windows dans le second.

252 J'ai utilisé un script perl pour agglomérer les mesures correspondant à une même opération, et enregistrer la suite de valeur dans une série de fichiers séparés. J'ai utilisé GNUPLOT pour transformer la série de données en graphiques postscript, en automatisant à l'aide d'un second script en perl. Enfin, j'ai utilisé un dernier script perl pour générer toutes les références aux

figures dans un fichier L^AT_EX.

254 Le format que j'utilise habituellement pour faire de la documentation est docbook. La chaîne de production docbook est la suivante : fichier docbook(xml), transformation à l'aide de XSLTPROC et d'une feuille de style en fichier FO (formatting objects), puis passage par FOP pour générer un fichier pdf ou rtf. Il s'avère que dans mon installation, la première étape se déroule correctement, mais la suivante ne se termine jamais lorsque le document référence des figures.

256 Le travail a été facilité par l'homogénéité des plateformes (même architecture, même système d'exploitation, même arborescence des fichiers dans mes répertoires. J'ai eu à configurer SSH au passage pour ne pas avoir de mot de passe à donner lorsque je me loge à distance.

258 C'est la chose la plus déroutante avec les applications Corba : elle sont en général relativement délicates à lancer : il faut démarrer dans le bon ordre et avec les bons paramètres le service de nom, les repository, etc. En revanche, une fois ce travail effectué localement, le fonctionnement des deux objets distants se passe de la même façon. Bien que ce soit le but du middleware, cette véritable transparence réseau est vraiment fascinante.

5.1.3 Commentaire sur les mesures

263 Lors de la consultation des graphiques, il faut prêter attention à l'échelle en seconde, situés sur le côté gauche. L'échelle dépend des variations extrêmes des grandeurs mesurées.

265 La mesure la plus intéressante est celle de l'opération Associate, qui est la première à être invoquée. On distingue nettement que le premier appel est assez long, tandis que les suivants sont assez constamment plus courts.

266 On peut supposer que quelque chose se met en route correspondant à une initialisation. Le quelque chose pouvant être la couche Corba, la couche réseau, ou un autre élément du système difficile à déterminer.

267 Autre hypothèse possible : que le client localise le serveur au moment du premier appel, et que son adresse est ensuite connue.

5.2 Pilotage des objets RTP : collaboration avec Nicolas Bouillot

271 Nicolas Bouillot travaille sur le transferts de flux dans le cadre du concert virtuel réparti. Il était donc naturel de tester "l'interopérabilité" de nos objets.

272 Le travail était rendu à la fois plus facile et plus difficile par le fait que nos `home` respectifs étaient désormais sur une partition NFS. La simplification venait du fait de ne plus avoir la réplication des fichiers à gérer, la

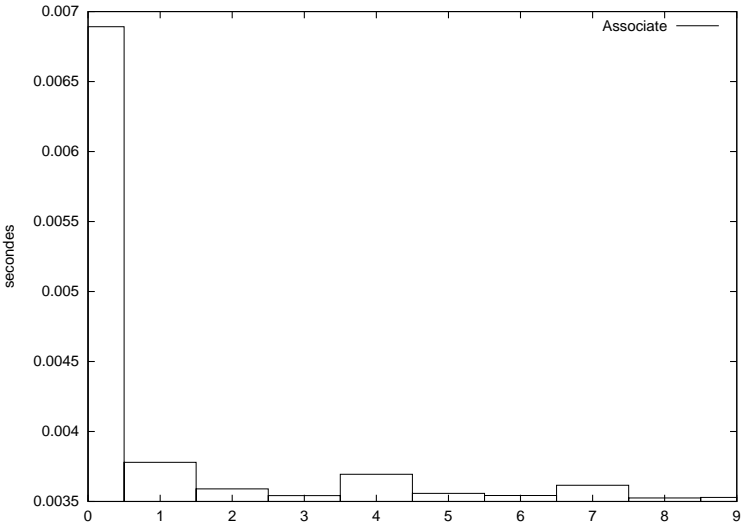


FIG. 10 – la fonction Associate mesurée localement

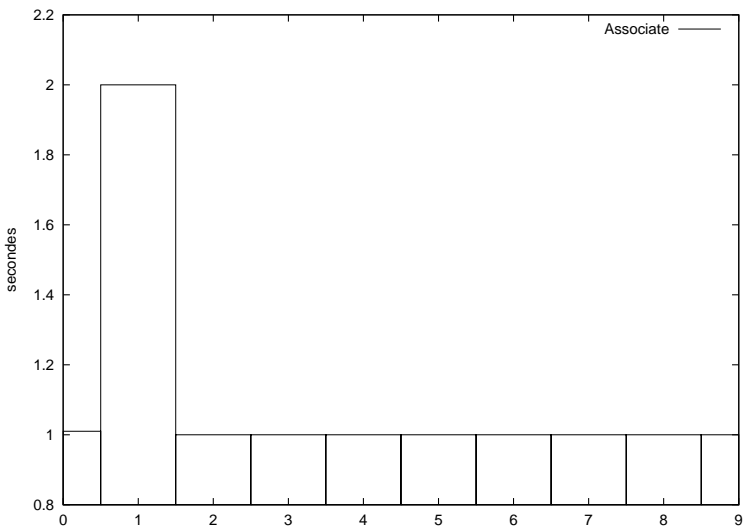


FIG. 11 – *la fonction Associate mesurée à distance*

complication venait du fait de devoir écrire des sections conditionnelles dans les fichiers de configuration `.bashrc`, `.jmaxrc`, etc.

273 Le travail était rendu facile par le rapprochement de nos bureaux (dans la même pièce), et par le fait que nous avons eu de nombreux échanges, notamment au sujet de l'utilisation de l'interface de programmation de JMAX, et de certaines idées de l'Ircam.

274 Les objets en eux-même ont fonctionné relativement rapidement. Une des difficulté est venue de NFS. Mon objet créait une socket `~locher/tmp/socket_moritaz_serveur`, qui reposaient donc sur le serveur, d'où collision lorsque l'on l'exécutait sur plusieurs machines en même temps !!

5.3 L'informatique au quotidien

5.3.1 Déménager

278 Le fait de devoir déplacer la chemise en carton contenant des notes et de la documentation n'est évidemment pas le plus grand problème. Ce qui l'est plus, c'est la changement de machine...

279 Evidemment grâce à SSH et la version sécurisée des r-tools (scp par ex), les transferts sont possibles. Mais en général lorsque l'on récupère une machine, il reste des fichiers des précédents utilisateurs, que l'on ne peut pas effacer. Il s'agit alors de pratiquer un jeu assez savant de chaise musicale, pour sauvegarder une partie sur une autre machine, récupérer la partie la plus cruciale de ses fichiers pour pouvoir travailler...

280 L'administrateur a des contraintes et n'intervient pas toujours au moment opportun.

281 On s'aperçoit assez vite que certains logiciels manquent... Je suis un utilisateur de emacs, et formate ma documentation en Docbook. Il me faut donc les outils adéquats sur mon poste de travail... les versions changent, les fichiers de configuration aussi...

Un serveur NFS *283* L'avantage parait assez évident : on se connecte à n'importe laquelle des machine et on retrouve ses fichiers.

284 L'inconvénient est une conséquence de l'avantage (cette unité), c'est le même fichier de configuration qui est lu sur chaque machine, ce qui impose l'écriture de sections conditionnelles. C'est sans grande difficulté en bash, c'est un peu plus délicat en tcl lorsque l'on ne s'en sert que pour le fichier `.jmaxrc`.

285 Il faut penser que l'inode de la socket sera la même pour chaque machine si l'on se situe sur le volume NFS, le fait de la mettre dans `/tmp` résoud le problème.

286 Enfin pour terminer dans la catégorie inconvénient, si le serveur NFS

n'est pas situé sur une machine dédiée, et que l'on tache d'y installer des drivers en version beta, la machine plante et tout le monde est bloqué.

287 Le fait de changer de répertoire complique un peu la configuration (MOZILLA et KDE stockent en dur le chemin vers les fichiers annexes), les chemins que l'on a laissé en dur dans les scripts.

288 Bien sur, il existe une parade à chacun des problème, et c'est un petit défi assez plaisant à chaque fois de la trouver... mais pendant ce temps ni le développement ni la rédaction n'avance.

289 Une chose est sûre, malgré les progrès sans fin de l'art informatique, il y aura toujours des administrateurs pour gérer les cas où "ça ne marche pas" !

5.3.2 Ne pas être root

294 Mon premier mois de stage se déroulait en juillet-août, au moment où l'administrateur était en vacance. Comme il me manquait quelques outils, il a fallu que je me débrouille sans le mot de passe du compte root, en apprenant à installer les binaires dans un repertoire \$HOME/bin, etc.

295 Cette configuration c'est révélée très pratique par la suite, lorsque le serveur NFS a été installé, puisque toute les bibliothèques nécessaires se trouvaient sur le volume réseau.

296 Actuellement, la pratique d'Unix a un peu changée et cela probablement en raison de la diffusion massive de GNU/Linux. L'utilisateur est très souvent l'administrateur de sa machine, et a l'habitude de cette autonomie. Dépendre d'un tiers pour la configuration de sa machine n'est pas évident. C'est probablement une source de bonnes habitudes, puisque cela oblige à bien séparer les besoins.

6 Travail de réflexion

6.1 Lectures des articles

302 Pour prendre le sujet en main, au début du stage, je me suis plongé dans la lecture (ou dans la relecture) d'articles liées au sujet.

303 Il s'est agit notamment des articles signés, ou co-signés, par Eric Gressier-Soudan, en charge du tutorat du stage.

304 Erwan Becquet, le développeur du logiciel OPENTAZ, a cosigné un certain nombre d'articles. (FIXME faire référence à la bibliographie tel article traite plutôt de ceci ou de cela...).

305 J'ai retrouvé, en suivant la bibliographie ou parfois juste au fil des recherches sur le Web, des articles des principaux acteurs du domaine, comme par exemple l'EPFL (FIXME lien vers la bibliographie).

6.2 Travail de compréhension

307 Le travail de compréhension pour des articles rédigés en langue anglaise comprend une part de traduction, même partielle. Cette démarche ne complique pas forcément la tâche, même si l'appréhension du fond de l'article est moins immédiate. Il est plus facile de lire en diagonale dans sa langue maternelle⁴. La lecture en langue étrangère est forcément plus attentive.

308 Les ouvrages normatifs (FIXME référence) sont particulièrement pénibles à consulter, et n'apportent pas grand chose tant que les grandes lignes restent floues.

309 Le User Guide de la KEMA (FIXME référence) a pour cela été un guide précieux. Il propose en première approche une description du fonctionnement de la norme ICCP en anglais, avant de commencer une description plus détaillée, avec des renvois nombreux aux textes normatifs.

311 (FIXME peut-être à déplacer dans la section développement ???)

312 Une séance d'explication a eu lieu lors de ma première semaine de stage. Erwan Becquet, m'a présenté son code et m'a expliqué l'ensemble du fonctionnement de son architecture. En particulier, il m'a présenté le mécanisme interne de gestion des événements, où l'ensemble des signaux, horloge ou modification de valeur sont traités de la même façon.

313 Cette rencontre a peut-être eu lieu un peu tôt pour moi, ma compréhension de la norme était alors balbutiante. Je ne suis cependant pas le seul à avoir un planning à organiser.

314 Les indications d'Erwan Becquet m'ont permis de savoir rapidement ce que je devais modifier dans les fichiers de configuration. Par la suite je me suis penché sur les outils d'autoconfiguration GNU (FIXME référence vers livre sur autotools), afin de mieux les comprendre.

315 Le fait de parvenir à compiler l'application a été ma première victoire!!!

316 Erwan Becquet a ensuite répondu très régulièrement aux mails que je pouvais lui envoyer.

319 En dehors de la compréhension de TASE.2, il me fallait découvrir également le fonctionnement de JMAX. De la documentation est disponible pour compiler, installer configurer l'application, mais la présence de Nicolas Bouillot m'a été précieuse.

320 Au delà, pour appréhender le fonctionnement du logiciel, le mieux est de charger un à un les items de l'aide en ligne de JMAX. Il n'y a pas, ou peu de tutoriaux disponibles du style "écrivons notre premier patch ensemble". La aussi, la présence de Nicolas Bouillot m'a permis de gagner du temps.

321 Le problème de la compréhension est celui du "bootstrapping", il faut bien tirer sur un premier fil avant de pouvoir s'accrocher aux autres.

322 Le fait de devoir programmer pour rapprocher les deux univers reste

⁴Contrairement à ce que mon nom et mon prénom pourrait laisser entendre, ma langue maternelle est le français.

cependant le principal moteur de la compréhension. Quelques soit la valeur *logicielle* de ce que j'ai écrits⁵, le fait de devoir concrétiser l'interfonctionnement des deux univers oblige à préciser l'idée que l'on se fait du fonctionnement de la messagerie ou de la plate-forme de programmation visuelle.

6.3 Adéquation des sémantiques

6.3.1 Liste des variables

326 TASE.2 a été conçu avec comme application typique la gestion d'une centrale thermique. Les objets à manager sont susceptibles de changer de temps en temps, mais possèdent une caractéristique : celle de la matérialité, qui s'inscrit dans la durée.

327 Le fait de concevoir un VCC comme une application configurée statiquement prend alors tout son sens. Le risque qu'un second bras pousse à un robot articulé pendant la nuit est nul.

328 Lorsqu'un nouvel objet est ajouté sur la plateforme, le jeu de variables est configuré, et au terme d'une adaptation plus ou moins longue, il est mis à disposition du réseau. Il suffit aux clients OPENTAZ de connaître le nom du nouvel Objet managé. Elle peuvent alors découvrir dynamiquement la liste de ses variables après avoir créé une association avec le nouveau VCC. Une fois la liste des variables accessibles en lecture et/ou en écriture connues, ces dernières sont consultables et/ou modifiables.

329 Un patch JMAX est un canevas souple pour permettre à son utilisateur de créer un *programme graphique* qui ressemble à ce qu'il veut. Il est par nature difficile, voire impossible, de connaître le nombre et la nature des points de contrôles.

330 La solution que je préconise, est de définir un certain nombre de jeux de variables typiques. Lorsque l'on crée un objet de contrôle, on peut alors lui affecter un **type** ou un autre selon les besoins. Cela donne un peu de flexibilité à l'utilisation, même si elle n'est pas suffisante.

332 En revanche, cela permet de donner à un point de contrôle (l'objet `mtaz_rc`) la granularité d'un serveur VCC, même si une présentation plus conviviale que de mapper toutes les variables sur les `inlets/outlets` serait possible.

333 FIXME Reprenons l'idée d'instrument virtuel, et donnons lui arbitrairement les propriétés suivantes. (FIXME pseudo-code). et suite exemple.

335 Comme cela à été vu lors de la soutenance avec François Dechelle, il faudrait que l'utilisateur du patch puisse définir des points de contrôle et que la console distante puisse découvrir les variables sous la forme :

```
nom_du_patch.variable
```

⁵Je ne pense pas être le plus à même d'en juger!!!

, ou, si l'on prend en compte le fait qu'il existe un objet implantant le pattern composite

`nomDuPatchPrincipal.`

ventuelsSousPatches

`*.nomVariable`

.
336 Le fait de devoir déclarer les points de contrôles avant d'écrire le patch est d'une lourdeur inadéquate, et il manque à TASE.2 la possibilité de changement dynamique de la liste des variables disponibles.⁶

6.3.2 Estampillage

338 Les différentes estampilles liées à la norme ne sont pas exploitées dans le prototype.

339 On dispose :

- . d'un estampille temporelle qui donne la date de production de la valeur
- . un champ décrit la confiance que l'on peut accorder au champ. (Variable produite ou calculée par exemple).
- . nombre de changements de valeurs depuis le dernier envoi d'un report

340 Il m'est difficile d'évaluer l'intérêt de ces données. Dans le cas d'un concert virtuel réparti, on peut s'y intéresser. Si une stratégie automatique est mise en place dans le client et qu'une politique de traitement est définie, les informations peuvent être utilisées.

341 Par exemple, si un mécanisme est mis en place pour synchroniser les différents protagonistes au travers de leurs VCC respectifs, le traitement pourrait ignorer des valeurs pas assez fraîches, ou augmenter la fréquence des reports si elle s'aperçoit qu'il y a trop de changement de valeurs (*COV*) entre deux Report.

⁶J'insiste sur le fait qu'il s'agit d'une lacune de la norme. Le code de OPENTAZ permettrait un comportement dynamique.

On pourrait alors imaginer une variable de service auquel le client s'abonnerait pour être notifié du changement de la liste des variables.

Ce comportement serait satisfaisant mais présente quelques inconvénients.

- . Il n'est pas normalisé
- . Il amalgame les variables de métadonnées et celles décrivant le fonctionnement de l'appareil piloté

342 D'autres besoins semblent malgré tout plus prioritaires.

343 Il peut être utile au client de connaître les bornes des variables numériques. Si le client doit calculer une interface graphique automatiquement, c'est même indispensable. Par exemple, si une variable est un pourcentage, les bornes iront de 0 à 100, et on pourra calculer la borne d'un (FIXME) potentiomètre graphique. Ici on peut le faire en définissant des variables supplémentaires en lecture seule.

344 Par exemple : pour une variable `volume`, il faudrait définir en lecture seule `volume_min` et `volume_max`. Et programmer une stratégie dans le client pour en tenir compte.

345 De nouveau l'inconvénient est que ce comportement n'est pas standard.

6.4 Interactions

347 Les différentes méthodes de report paraissent très adaptées. (TODO compléter).

348 L'envoi d'un **Report** lors du changement de valeur d'une variable donne une sémantique événementielle idéale pour le contrôle distant, tandis que les **Report** périodiques sont adaptés au monitoring.

349 (FIXME réécrire) Dans le cas du concert virtuel réparti, il y aurait par exemple un "chef d'orchestre" supervisant les différents flux audio, et les participants, avec un patch JMAX adapté, comprenant un VCC, et éventuellement un client faisant du monitoring pour savoir ce que font les autres.

6.5 Modèle objet

352 Le fait de pouvoir transmettre des objets plus complexes paraît intéresser l'IRCAM. Je parlerais plutôt de type composé pour éviter la confusion avec la POO.

353 Les applications seraient par exemple de pouvoir envoyer une série de notes pour les faire jouer par un à l'instrument virtuel distant.

354 L'objet en tant que sous-système d'un autre système paraît intéressant pour la granularité du contrôle. Par exemple une liste d'objet, qui serait elle-même un objet.

355 Les besoins tels que le support de l'héritage ne paraissent en revanche pas nécessaire. Les autres logiciels de supervision ne cherchent d'ailleurs pas du tout à proposer au contrôle distant les notions d'héritage.

356 Comme on agit sur des instances, on dispose de toutes ses propriétés, sans savoir si elle sont hérités ou pas.

6.6 L'avenir

358 CORBA ?

359 - découverte dynamique d'if

360 - lien de composition

361 - éventuellement même héritage

362 - lecture des propriétés avec connaissance des types

363 - invocation de méthodes

7 Travail de présentation

7.1 confection du rapport et rédaction intermédiaire

371 J'ai réalisé plusieurs documents au cours de ce stage. Un premier, simplement intitulé *Premier rapport* décrivait le fonctionnement de la norme TASE.2.

372 La séance de mesures a également donné lieu à la rédaction d'un rapport, comprenant les graphiques tirés des mesures. C'est le premier essai que j'ai du réaliser en \LaTeX .

7.1.1 Le choix des outils d'édition

374 Lors de mes travaux de rédaction pour mes projets d'études, j'ai toujours préféré la technique du texte compilé.

375 Suite à un article paru dans Linux-Magazine (FIXME référence), je m'étais intéressé à XSLT et XSL-FO. J'avais conçu une DTD pour rédiger mes petits documents.

376 Celle-ci c'est révélé rapidement insuffisante, et je me suis orienté vers Docbook, qui est de loin la DTD SGML la plus complète pour l'édition technique.

377 Je travaillais initialement avec l'environnement SGML, à savoir Jade et des feuilles de styles DSSSL.

378 La DTD a été portée en XML, et c'est maintenant celle de référence. Je me suis donc installé un processeur XSLT `XSLTPROC`, et `FOP` qui permet de convertir le format XSL-FO en pdf. Suite à des problèmes de configuration, j'ai rencontré quelques difficultés au moment de rédiger le document comprenant les résultats des mesures.

379 Un peu de travail plus tard, tout fonctionnait pour le mieux, mais cela n'a pas duré. La bataille Docbook contre \LaTeX faisait rage dans le bureau des stagiaires, et soyons humble, je l'ai perdue.

380 En effet, `FOP` a rendu l'âme après qu'un peu trop d'images aient été incorporées dans le document.

381 J'ai du me rabattre sur \LaTeX la dernière semaine de stage, entraînant

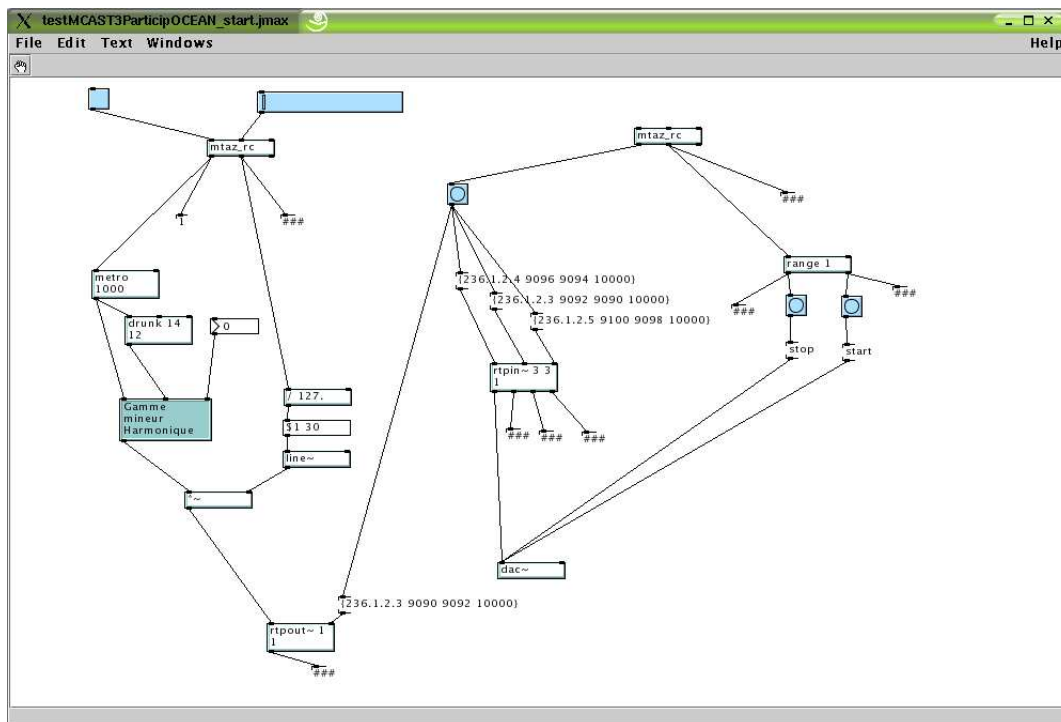


FIG. 12 – Le patch mettant en oeuvre `mtaz_rc` tournant sur chacune des machines

la rédaction d'un script en perl. Nicolas Bouillot m'a été d'une grande aide en me donnant la traduction des tags que j'utilisais en Docbook.

7.2 Préparation à la soutenance du stage

7.3 Démonstration avec trois machines devant François Dechelle

386 Une présentation a eu lieu devant François Dechelle, Eric Gressier-Soudan.

387 François Déchelle dirige l'équipe logiciels libres et ingénierie logicielle, récemment créée à l'IRCAM, après avoir dirigé l'équipe Systèmes temps réel.

388 Eric Gressier-Soudan, est membre du laboratoire CEDRIC/CNAM et enseignant au CNAM.

389 Le travail a été présenté conjointement avec Nicolas Bouillot.

390 La présentation mettait en oeuvre les objets `rtp_in` et `rtp_out` utilisant le multicast, et l'objet `mtaz_rc`.

393 Un patch conçu pour une précédente présentation sans le contrôle dis-

tant a été récupéré et complété. Celui-ci comporte un assemblage d'objets JMAX qui joue des notes aléatoires dans une gamme. Un objet de contrôle a été inséré entre le curseur de volume et la boîte à cocher mettant le son en route.

394 Les objets de Nicolas Bouillot sont paramétrés par des objets `messconst`, ou message constant. Il faut usuellement cliquer dessus pour que le message soit envoyé. Ici, les différents objets de contrôle ont été attachés à un second objet de contrôle, afin de pouvoir commander l'initialisation à distance.

395 De même le `dac~`, représentant la sortie de la carte son peut accepter le message "start" ou "stop". Le lien avec le second objet de contrôle permet de démarrer à distance les échanges de flux entre objets et la sortie vers la carte son. Il faut juste *filtrer* la valeur numérique envoyée, en envoyant un déclenchement sur le message "stop" si le nombre est 0 et "start" sinon.

396 L'ensemble des exécutable sont lancés par un script de ma confection qui réalise les appels de commande avec SSH pour les machines distantes.

397 (FIXME extrait du script)

398 Certains détails pratiques doivent être envisagés. KDE utilise un serveur de son qui ouvre les périphériques son. Le premier utilisateur à créer une session prend les droits sur les périphériques audio. Il faut donc veiller à ce qu'aucun autre utilisateur n'ai pris les droits sur les périphériques. Par ailleurs, avec SSH la variable d'environnement `DISPLAY` est automatiquement redéfinie vers l'affichage réel de celui qui lance les commandes, et il faut forcer sa valeur vers la machine qui exécute réellement les commandes. Si on lance l'exécution de JMAX sur *ppc1* depuis *oceanonix*, par défaut la fenêtre s'affichera sur *oceanonix*. Il faut donc corriger ce comportement. Une commande

```
xhost +$(hostname)
```

est indispensable, les droit d'affichage n'étant pas donné aux sessions distantes par défaut.

399 M. Dechelle a paru satisfait de la démonstration. Le besoin de contrôle distant est le suivant : le créateur du patch doit pouvoir définir des points de contrôle accessibles depuis la console (le client OPENTAZ), et cette dernière doit pouvoir les découvrir dynamiquement. Ces points sont satisfaits par l'application. Le fait de pouvoir transférer des objets un peu plus complexes, comme une suite de notes de musique est également formulé, et paraît plus dur à satisfaire, et implique éventuellement le passage à d'autres technologies.

8 Conclusion : Les acquis du stage

8.1 Ma pratique du développement

403 Ma pratique du développement se résume au travaux pratiques et autres projets auxquels j'ai eu l'occasion de participer dans le cadre des études au CNAM, ainsi que des développements personnels, liées essentiellement à la maintenance d'une page Web.

404 Les projets m'ont permis de travailler, entre autres, avec les langages Ada, java, C++ et d'utiliser CORBA.

405 C'est dans ces travaux personnels que j'ai eu l'occasion d'apprendre à travailler avec XML en utilisant les transformations XSL, et d'utiliser java, et plus largement d'utiliser et de configurer GNU/Linux.

406 Mon activité professionnelle n'a pas de lien direct avec l'informatique, du moins pour le moment. L'avenir du cinéma est lié à l'informatique, au titre de la convergence actuelle des médias vers le numérique. Actuellement, l'informatique intervient dans mon travail pour l'automatisation des démarrages et des changements d'éclairage et de volume sonore pendant les séances.

8.2 Faire simple

409 Ma vision idéalisée de l'informatique me pousse toujours à envisager des logiciels riches en fonctionnalités et donc complexes. Étant en général mon propre client, je peux me permettre ce perfectionnisme, qui est plutôt un défaut.

410 Il faut parfois vraiment raffiner pour concevoir quelque chose de simple. La question "quelles sont les quelques fonctionnalités vraiment essentielles dans ce que je veux faire?" n'a pas une réponse évidente.

411 C'est ce à quoi ce stage me sert. A faire un programme qui ne fait rien d'extraordinaire, mais qui fonctionne. Il m'a fallu un peu de temps pour renoncer à toutes les idées que j'avais eu au préalable, mais qui rendait le projet irréalisable.

413 Un premier travail de conception, pas entièrement formalisé, m'a permis de trouver une direction. Je me suis rendu compte que mes brouillons successifs devenaient stériles, dans la mesure où je n'avais pas assez en main la matière que je devais utiliser, à savoir le code des deux programmes à interfacier. J'ai mieux compris à travers ce stage ce que "développement incrémental" voulait dire.

414 Je me suis permis pour ce travail de mettre le nez dans les sources en codant directement. Je me suis même obligé à le faire. Cette attitude oblige à bien comprendre le code sur lequel l'on travaille, et qui est celui de quelqu'un d'autre, même si les aller-retour essai-erreur peuvent prendre du temps.

415 Le travail de conception devient possible, pour préparer, par exemple, une seconde version plus solide, et qui est l'objet de ce document.

8.3 L'expérience en laboratoire

418 Le travail en laboratoire laisse une grande part d'autonomie dans l'organisation du travail.

8.4 Quelques contraintes d'organisation

8.4.1 Une ressource unique : la clef

421 Pour pouvoir se rendre au bureau des stagiaires, il faut accéder à une clef, qui est conservée par défaut par le gardien de l'accès rue de Conté du CNAM. Il faut être inscrit sur la liste des détenteurs possibles de la clef. Le premier arrivé retire la clef, et ouvre ensuite aux autres. Tout se passe bien dans l'ensemble, mais il y a parfois quelques ratés.

422 Les parallèles informatique possibles sont le sujet de nombreuses plaisanteries. Ressource unique, cohorte, liste bilatérale des droits...

9 Remerciements

426

Références

- [1] *Component oriented control architecture : the COCA project*. Microprocessors and microsystems, mar 1999.
- [2] *Enhancing numerical controllers, using MMS concepts and a CORBA-based software bus*, volume 14, 2001.
- [3] *Towards a RT-Java Embedded Remote Monitoring Tool for Small and Medium Power Plant Units*. ETFA, 2001.
- [4] François Dechelle and Norbert Schnell. *Developing jMax control objects*. IRCAM, Paris, 1999. documentation en ligne.
- [5] François Dechelle and Norbert Schnell. *Developing jMax DSP objects*. IRCAM, Paris, 1999. documentation en ligne.
- [6] Fedor, Schoffstall, and Davin. *A Simple Network Management Protocol*, may 1990.
- [7] FeT. *Object Oriented timed messaging service for industrial ethernet : a fieldbus like architecture for powerplant control and factory automation*, 2001.

- [8] Patrick Kadionik. Administrez facilement votre réseau : Snmp. *GNU/Linux Magazine France*, (43), oct 2002.
- [9] Bouillot Nicolas. Métaphore de l'orchestre virtuel, étude des contraintes systèmes et réseaux, prototypage. Rapport intermédiaire de thèse, 2002.
- [10] OMG. *Real Time CORBA-MMS for Embedded Systems*, San-Diego, 2000.
- [11] Patrick Pleine. Des réseaux à tout faire ou presque. *Flash Informatique*, (6), jun 1994.
- [12] Saxton, Ambrose, and Kendall. *ICCP User Guide*, oct 1996.
- [13] Luc Teboul. *Programmation par aspect appliqué à une messagerie industrielle*. PhD thesis, Université Paris 6 et CNAM/CEDRIC, 2002.
- [14] Utility Communication Specification Working Group. *TASE.2 Object Models IEC 870-6-802*, 1997.
- [15] Utility Communication Specification Working Group. *TASE.2 profiles IEC 870-6-702*, 1997.
- [16] Utility Communication Specification Working Group. *TASE.2 Services and Protocols IEC 870-6-503*, 1997.
- [17] WFCS. *Prototyping QoS based architecture for Power Plant Control Application*, 2000.