Remerciements

Je tiens vivement à remercier Roger Pissard-Gibollet, responsable du service Support Expérimentation et Développement de l'INRIA Rhône-Alpes, de m'avoir accueilli et d'avoir mis à ma disposition tous les moyens nécessaires au bon déroulement de ce projet de fin d'étude.

Je remercie également Sébastien Jarde, mon maître de stage, qui a suivi et encadré mon travail. Sa très bonne connaissance de la plate-forme BIP et ses conseils ont grandement facilité mon travail.

Je remercie Soraya Arias pour sa rigueur et ses nombreux conseils au sujet du développement logiciel.

Je remercie Olivier Richard responsable des stages RICM3 option réseaux.

Je remercie également l'ensemble des membres du service S.E.D, notamment G. Baille, J.F Cuniberto, H Mathieu pour leur accueil, leur gentillesse et leur disponibilité.

Enfin, je remercie mes compagnons de stage :Abdellah Eljalaoui, Jérôme Fuchet, Julien Mélot et Lucile Prin-Zanet qui ont créé une ambiance chaleureuse tout au long de mon séjour.

Développement d'un superviseur pour plates-formes Robotiques

Résumé

Le service Support Expérimentation et Développement (SED) de l'INRIA Rhône-Alpes assure la mise en place et le support des plates-formes robotiques pour les expérimentations des projets de recherche. J'ai réalisé mon stage de fin de cycle ingénieur au sein de ce service, consacrant cette période de six mois à la conception d'un logiciel de supervision. Ce logiciel de type client-serveur permet à ses utilisateurs d'observer et d'agir sur le contrôle de plates-formes en fonctionnement. Ce rapport aborde donc dans sa première partie, la conception et la réalisation d'un logiciel générique aux plates-formes de l'INRIA. Il décrit ensuite la mise en place du superviseur sur une plate-forme spécifique : le robot BIP, un robot anthropomorphe, a 15 degrés de liberté.

Mots clef : JAVA, CLIENT-SERVEUR, PROXY, XML, MONITORING DE DONNEES, SUPERVISION, PLATES-FORMES ROBOTIQUES

Designing of a supervision software

Abstract

I carried out during my six month internship in the "Support Expérimentation et Développement" service (SED) at INRIA Rhône-Alpes. This departement ensures the installation and the robotic platform support for the research project experiments. I was in charge of designing a supervision software. This client-server software allows the users to watch and conrol a robotic system under operation. This report focuses firstly on the design and the realization of a generic software for the INRIA robotic systems. Then it describes the installation of the supervisor on a specific platform : BIP robot, an anthropomorphic walking robot with fifteen degrees of freedom.

Key Word : JAVA, CLIENT-SERVEUR, PROXY, XML,DATA MONITORING, SUPERVISION, ROBOTIC PLATFORMS

Table des matières

Table des figures

Première partie

Introduction

Chapitre 1

Contexte du stage

1.1 Présentation de l'INRIA

Créé en 1967 à Rocquencourt près de Paris, l'Institut National de Recherche en Informatique et Automatique (INRIA), est un établissement public à caractère scientifique et technologique (EPST) placé sous la double tutelle du Ministère de la recherche et du Ministère de l'économie, des finances et de l'industrie. Cette institut mène des recherches avancées dans le domaine des sciences et technologies de l'information et de la communication. Ce domaine inclut l'informatique et l'automatique, mais aussi les télécommunications et le multimédia, la robotique, le traitement du signal et le calcul scientifique. L'INRIA est composé de 6 unités de recherche en France.

1.1.1 Quelques chiffres

Les chiffres énoncés ci-dessous datent de janvier 2003.

- Ressources budgétaires
  - Budget global : 120 M Euros HT
  - Ressources propres : 1/4 du budget global
- Ressources humaines : 3000 personnes
  - 900 titulaires INRIA (400 chercheurs, 500 ingénieurs et techniciens)
  - 750 post-doctorants, stagiaires, invités.
  - 700 doctorants.
  - 450 chercheurs et enseignants d'autres organismes.
  - 200 "ingénieurs experts" (sur contrat de recherche).
- Indicateurs
  - Plus de 600 contrats de recettes actifs.
  - 300 contrats de recettes signés dans l'année.
  - Une soixantaine de sociétés sont issues de l'INRIA

- L'INRIA disposait en 2002 de 47 brevets prioritaires et maintenait au total 174 brevets (prioritaires + extensions).
- 60 licences payantes de logiciels et 60 licences de logiciels étaient actives en janvier 2002.
- Plus de 143 logiciels sont disponibles en accès gratuit sur le site de l'INRIA ou à travers la diffusion d'un CD-ROM.

1.1.2 L'INRIA Rhône-Alpes

L'unité Rhône-Alpes a vu le jour en 1992 et regroupe environ 470 personnes. 370 scientifiques sont répartis dans 25 équipes (17 projets de recherche), dont 16 sont en partenariat avec le CNRS, l'université Joseph-Fourier et l'institut national polytechnique de Grenoble(laboratoire GRAVIR et ID), l'école nationale supérieure de Lyon (laboratoire LIP), l'université Claudes-Bernard et l'Institut national des sciences appliquées (INSA). Six équipes de recherche sont totalement ou partiellement localisées à Lyon.

Elle mène ses activités en étroite collaboration avec les laboratoires de recherche publics et privés, nationaux et internationaux, et elle entretient des liens privilégiés avec l'institut d'Informatique et Mathématiques Appliquées de Grenoble (IMAG). Ces activités sont organisées autour de quatre pôles de recherche divisés en plusieurs équipes autonomes :

1. Maîtriser les systèmes et réseaux informatiques (Réseaux, parallélisme et systèmes répartis).
   Équipes : APACHE, ARENAIRE, ARES, COMPSYS, PLANETE, ReMaP, RESO, SARDES, TRIO, VASY, POPART.
2. Aider à la conception et à la création (Bases de connaissances, documents multimédia, modèles cognitifs).
   Équipes : EXMO, HELIX, I3D, WAM, PRIMA
3. Percevoir, simuler et agir (Synthèse d'images, réalité virtuelle, vision par ordinateur et robotique).
   Équipes : EVASION, ARTIS, MOVI, CYBERMOVE.
4. Modéliser les phénomènes complexes (Automatique, simulation et calcul scientifique).
   Équipes : IDOPT, IS2, BIPOP, OPALE.

1.1.3 Le service S.E.D

La mission du service Support Expérimentation et Développement est d'assurer le support aux plates-formes robotique expérimentales, vision et réalité virtuelle. Il est chargé de :

- Maintenir et assurer le support des systèmes expérimentaux(matériel et logiciels spécialisés)
- Développer des systèmes expérimentaux (mise en place d'expérimentations, logiciels)
- Participer à la recherche (conception de systèmes, participation aux expérimentations)

Le but est de favoriser :

- Les expérimentations inter-projets,
- La mise en commun des moyens expérimentaux,
- Les outils réutilisables (environnement de développement, matériel de réalité virtuelle..).

Chapitre 2

Problématique

Les laboratoires de recherche en robotique utilisent des plates-formes robotiques expérimentales pour valider et illustrer leurs travaux de recherche. A l'INRIA Rhône-Alpes, le support de ces plates-formes est assuré par le service S.E.D [26]. Le déroulement des expérimentations est assez similaire quelque soit la plate-forme (robot marcheur, robot mobile, vision ...). Il faut préparer l'expérience, la réaliser puis analyser son déroulement. Même si ce principe parait, simple il peut vite devenir très complexe avec des systèmes robotiques très évolués. C'est pourquoi, il apparru nécessaire d'utiliser un système qui permet de superviser le déroulement des manipulations sur ces plates-formes robotiques.

Chapitre 3

Intérêt du stage

Le stage que j'ai réalisé à l'INRIA Rhône-Alpes au sein du service S.E.D, a pour but l'étude des systèmes de supervision pour les plates-formes robotiques expérimentales. Il doit proposer une solution de supervision pour l'ensemble des plates-formes du service. Le travail effectué durant ce stage d'une durée de six mois s'organise en deux parties :

1. ***Analyse et conception d'un superviseur pour les plates-formes expérimentales***
2. ***Mise en place du système de supervision sur la plate-forme expérimentale BIP***

La première partie du stage a essentiellement porté sur la définition des besoins auxquels doit répondre un système de supervision. Avec l'aide des autres membres du service j'ai défini un cahier des charges et les spécifications qui en découlent. J'ai ensuite étudié les solutions déjà existantes. Cette étude m'a permis de découvrir et de mieux appréhender la problématiqe liée à la supervision. J'ai ensuite établi une architecture qui réponde au mieux aux besoins.

La deuxième partie de mon stage m'a permis de développer le superviseur et de le metrre en place pour la plate-forme BIP. Durant cette étape qui a occupé la plus grande partie de mon temps, j'ai concu et réalisé toute l'infrastructure nécessaire à la supervision du robot.

Nous arborderons dans la première partie du rapport le contexte du stage. Dans la seconde partie, nous arborderons les principes de la supervision robotique, puis le cahier des charges et les spécifications du logiciel de supervision. Nous finirons par la conception de ce logiciel écrit en JAVA. La troisième et dernière partie décrit la mise en place du logiciel de supervision sur la plate-forme BIP.

La partie suivante présente les principes de la supervision de plates-formes robotiques expérimentale.

Deuxième partie

La supervision de plates-formes robotiques expérimentales

Chapitre 4

La supervision des plates-formes

4.1 Qu'est-ce qu'une plate-forme robotique expérimentale

Une plate-forme robotique expérimentale est une machine de complexité variable composée d'une structure mécanique et d'une chaîne électromécanique. La partie mécanique assurant le déplacement est très variée : cela va du bras articulé à un mobile à roues en passant par des systèmes marcheurs.

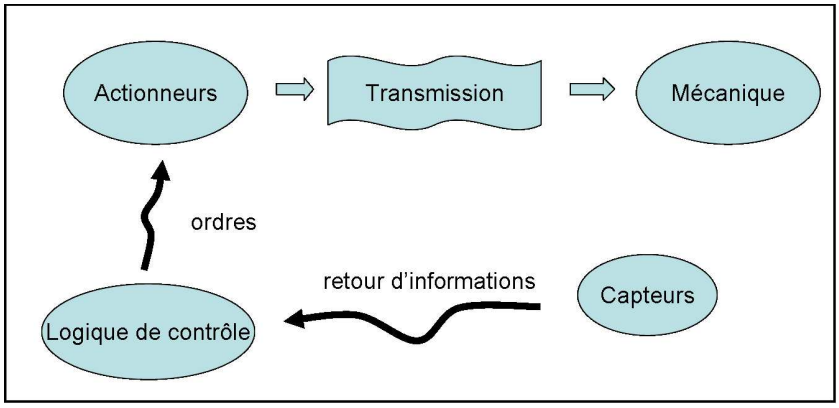

FIG. 4.1 – Composition d'une plate-forme robotique expérimentale

Comme on peut le voir sur le schéma (cf figure 4.1) la partie mécanique est mise en mouvement par les actionneurs via des transmissions. Les actionneurs sont eux-mêmes commandés par une logique de contrôle qui prend des décisions en fonction des informations en provenance des capteurs et des nombreux autres paramètres spécifiques à chaque plate-forme.

4.2 Comprendre la supervision robotique

Mon étude se focalise sur la supervision de plates-formes robotiques et non pas la supervision de processus industriels, (sujet beaucoup plus vaste). Les systèmes que l'on envisage de superviser sont des process :

- discrets et non pas continus
- sur lesquels on connaît les phases de déroulement
  1. initialisation
  2. expérimentation
  3. arrêt de l'expérience
  4. analyse des données

Définition de la supervision robotique : procédé à travers lequel un utilisateur peut observer et agir sur le contrôle d'une plate-forme robotique en fonctionnement.

La supervision d'une plate-forme robotique repose sur deux tâches principales.

- Le contrôle d'exécution.
- Le monitoring des données.

Nous allons préciser dans ce qui suit ces deux fonctions.

4.2.1 Le contrôle d'exécution

Toute expérimentation nécessite une phase d'initialisation qui consiste à mettre le système robotique dans un état initial tel que la manipulation puisse démarrer. A ce stade, il faut souvent charger et exécuter des programmes, mettre la mécanique dans une position définie... Ensuite une fois la plate-forme initialisée, il faut choisir la manipulation à lancer, l'exécuter, attendre la fin de son exécution ou l'arrêter d'urgence s'il y a un problème. L'ensemble de ces actions constitue le contrôle d'exécution.

4.2.2 Le monitoring des données

Lors de l'expérimentation, la tâche de monitoring des données consiste à visualiser et enregistrer l'état du système robotique. Cet état est constitué par les valeurs des capteurs **proprioceptifs** (données internes du robot telles que les positions articulaires) et des capteurs **extéroceptifs** (données de l'environnement telles que les capteurs de distance, visuel etc.) et des variables logicielles issues des contrôles fait sur le robot. Si on prend comme exemple un robot bipède, il est important de connaître les positions des articulations, les forces exercées sous les pieds... Ces données pourront être analysées pour déterminer si le robot reste en équilibre ou s'il va tomber, par exemple.
Ainsi le monitoring des données se décompose en 2 fonctions :

- La récupération des données :
  Le système de supervision doit pouvoir interroger la plate-forme de manière à obtenir l'ensemble des données importantes aux yeux de l'utilisateur qui mène l'expérience. On appelle cet ensemble de données **la télémétrie**.
- La représentation des données télémétriques :
  Le système de supervision a aussi la charge de la représentation de la télémétrie. Que ce soit de manière graphique, textuelle, vocale ou par tout autre moyen, le but est de fournir à l'utilisateur une représentation compréhensible de ces informations.

Il existe dans l'industrie spatiale des logiciel de supervision tel que "DREAM". La société TRASYS qui a réalisé ce locigiel l'a mis à disposition du service SED pour une évaluation. Le chapitre suivant présente de manière exhaustive cette évaluation.

Chapitre 5

Un exemple : le logiciel DREAM

Certaines des informations de cette section ne sont pas exhaustive car je ne disposais pas des documentations sur le logiciel DREAM.

5.1 Présentation

Dans le but de préparer les futures missions robotiques automatisées de l'Agence Spaciale Européenne[21], le projet JET a été créé. Son but principal est de valider les différentes technologies de monitoring et de contrôle de plates-formes robotiques. Riche des connaissances acquises lors des précédents projets et particulièrement le projet VIABLE[23], la société TRASYS[22] a développé le logiciel DREAM "Distributed Robotics and Automation Environment for Advanced Missions Specification and Supervision". Ce logiciel "tout en un" permet de superviser les différents outils robotiques de l'ESA. Présenté lors de la conférence ASTRA2000, il utilise les dernières technologies telles que JAVA et le langage robotique PDL [1].

5.2 Organisation logiciel

Le logiciel DREAM se compose de plusieurs modules qui prennent en charge toutes les phases de l'utilisation d'un système robotique. Il est capable de gérer toutes les plates-formes robotiques de type COMAU [24].

- **phase de préparation**
  Un module permet de décrire dans le langage PDL2 une manipulation à réaliser. Ce langage de programmation est dédié aux robots COMAU. Le système génère ensuite une application de contrôle de la plate-forme robotique.

- **phase de vérification**
  Un module de simulation des différentes plates-formes permet de valider le fonctionnement de l'application précédemment construite.
- **phase d'utilisation**
  Un module permet de charger une application sur une plate-forme et de suivre son bon déroulement via un système de contrôle d'exécution et de supervision de données.
- **phase d'analyse**
  Un module permet de rejouer une application pour analyser son déroulement. Ce module utilise une base de données qui stocke toutes les manipulations déjà réalisées.

5.2.1 Monitoring des données

Le module M2D se charge de la représentation des données de télémétrie. Un fichier de configuration permet de définir le contenu de cette télémétrie et un autre permet de définir sa représentation visuelle.

Voici une illustration de la gestion du monitoring sous DREAM :

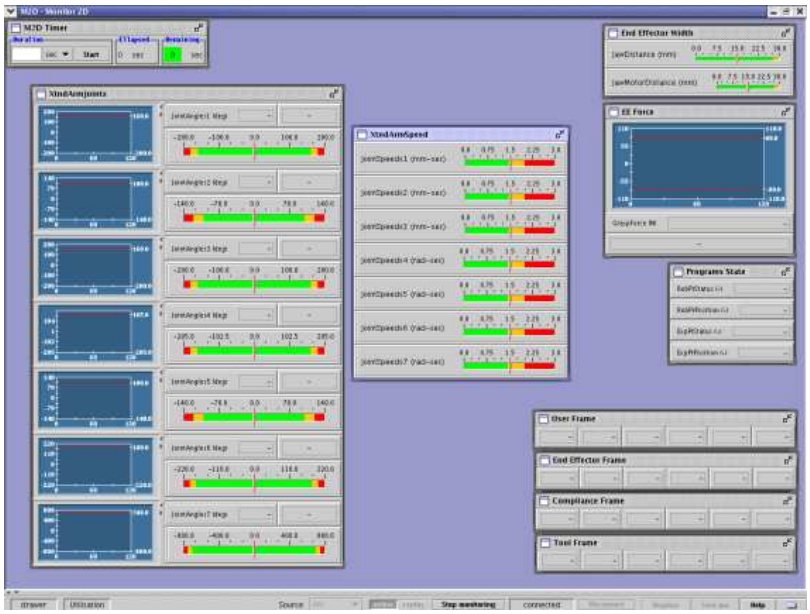

FIG. 5.1 – Un exemple de monitoring sous Dream

Comme on le constate la représentation des données de télémétrie peut se faire sous plusieurs formes :

- Graphique,
- Jauge,
- Valeur instantané.

5.3 Analyse du logiciel

Une partie de mon stage a été consacré à l'analyse de ce logiciel. C'est un logiciel assez gourmand en ressource CPU, en effet il se compose d'une bonne dizaine de processus et occupe pas moins de 40 Mo de mémoire vive. Son interface est complexe, et dispose de nombreux modules. Sa configuration à base de fichiers XML lui confère une souplesse et une adaptabilité remarquable notamment pour la partie de monitoring des données. Il est très facile de modifier la représentation des données de télémétrie dans l'IHM. L'adaptation à une nouvelle plate-forme ne pose pas de problème particulier. Par contre, il impose l'utilisation du langage PDL pour le développement des applications. De ce point de vue, il semble peu adapté a d'autres plates-formes qui n'utilisent pas cette méthode de développement. Cette partie semble plus spécifique aux robots COMAU de l'ESA.

En conclusion, même si ce logiciel est très complet et performant son utilisation dans un cadre autre que l'exploitation des plates-formes ESA semble difficile. Par contre la partie monitoring des données très puissante, peut très facilement être réutilisée comme nous le verrons dans la section 9.5.3

Ce logiciel de supervision n'est donc pas utilisable tel quel pour les plates-formes de l'INRIA. Le chapitre suivant présente les besoins spécifiques de l'INRIA. Il faut donc envisager de construire un systèmre plus approprié aux besoins spécifiques de ces plates-formes. La partie suivante présente ces besoins.

Chapitre 6

Supervision des plates-formes robotiques de l'INRIA Rhône Alpes

6.1 Définition des besoins

Les plates-formes robotique du service S.E.D [26] de l'INRIA fonctionnent sur le principe exposé sur la figure 4.1. Actuellement, les expérimentations sur ces plates-formes et l'exploitation des résultats sont très complexes. Chacune d'elle a son propre fonctionnement avec des outils dédiés. Parfois, il faut utiliser de nombreux scripts de configuration et des outils non graphiques. Aussi leur utilisation n'est pas aisé car il faut parfaitement les connaître. Dans un soucis de simplification et d'uniformisation de leur utilisation, le service S.E.D a besoin d'un système de supervision qui soit capable de gérer l'ensemble de ses plates-formes robotiques.

FIG. 6.1 – Les plates-formes robotiques du S.E.D

6.2 Cahier des charges et spécifications

Les besoins de supervision des plates-formes de l'INRIA Rhône-Alpes sont les mêmes que pour toute plate-forme en général. A savoir la visualisation des données et le contrôle de l'exécution.

6.2.1 Visualisation des données

Lors du déroulement d'une expérience, la plate-forme génère un ensemble de données d'exploitation **(la télémétrie)**. Le superviseur doit pouvoir récupérer et représenter ces informations. Cette représentation doit être possible en ligne ou hors ligne et avec ou sans historique.

Ces données se répartissent en trois classes à savoir :

- **Les états des capteurs :**
  Une plate-forme dispose de capteurs qui lui fournissent des informations sur l'environnement : capteur de positions (infra rouges ultra-sons...), de force, de température ...
- **Les états actionneurs :**
  Ce type de données fournit des informations sur l'état interne des systèmes tels que la position d'articulations, positions des moteurs, tensions de fonctionnments...
- **Les alarmes :**
  Dans le cas d'un mauvais fonctionnement ce type de donnée peu fournir la raison du dysfonctionnement sous forme de message d'erreur.

6.2.2 Contrôle de l'exécution

Le système de supervision doit offrir la possibilité de contrôler la plate-forme à distance. Cela signifie que l'utilisateur peut contrôler le déroulement des manipulations sur la plate-forme sans une intervention directe. Le superviseur doit offrir les possibilités suivantes :

- Charger une application.
- Lancer, stopper l'application.
- Mettre en pause le déroulement de l'application courante.
- Afficher l'état de l'application : arrêté, en initialisation, en exécution, fin d'exécution ...

6.2.3 Gestion de l'historique

Toutes les données recueillies lors des manipulations doivent être sauvegardées. Les fonctionnalités offertes à l'utilisateur sont :

- Recharger des manipulations dans le superviseur
- Rejouer des manipulations dans le superviseur
- Sauvegarder des résultats et des manipulations pour les exploiter a posteriori.

6.2.4 Contraintes sur les plates-formes robotiques

Les contraintes sur les plates-formes robotiques sont nombreuses :

- Faibles Ressources : les cartes embarquées ont des CPU de faible puissances, des cartes réseaux assez lentes etc.
- Contraintes temporelles : les système robotiques de l'INRIA imposent des impératifs temporels pour assurer leur bon fonctionnement (par exemple fournir une position articulaire toutes les 10ms).
- Utilisation partagée : les plates-formes doivent être accessibles à plusieurs utilisateurs simultanément.

6.3 Architecture globale

Afin de rendre les plates-formes robotiques accessibles à plusieurs utilisateurs simultanément, il a fallu choisir une architecture ouverte qui autorise les connexions multiples sur le système. L'architecture retenue, de type **clients-serveurs**, découle de cette nécessité et des besoins définis au paragraphe 6.2.

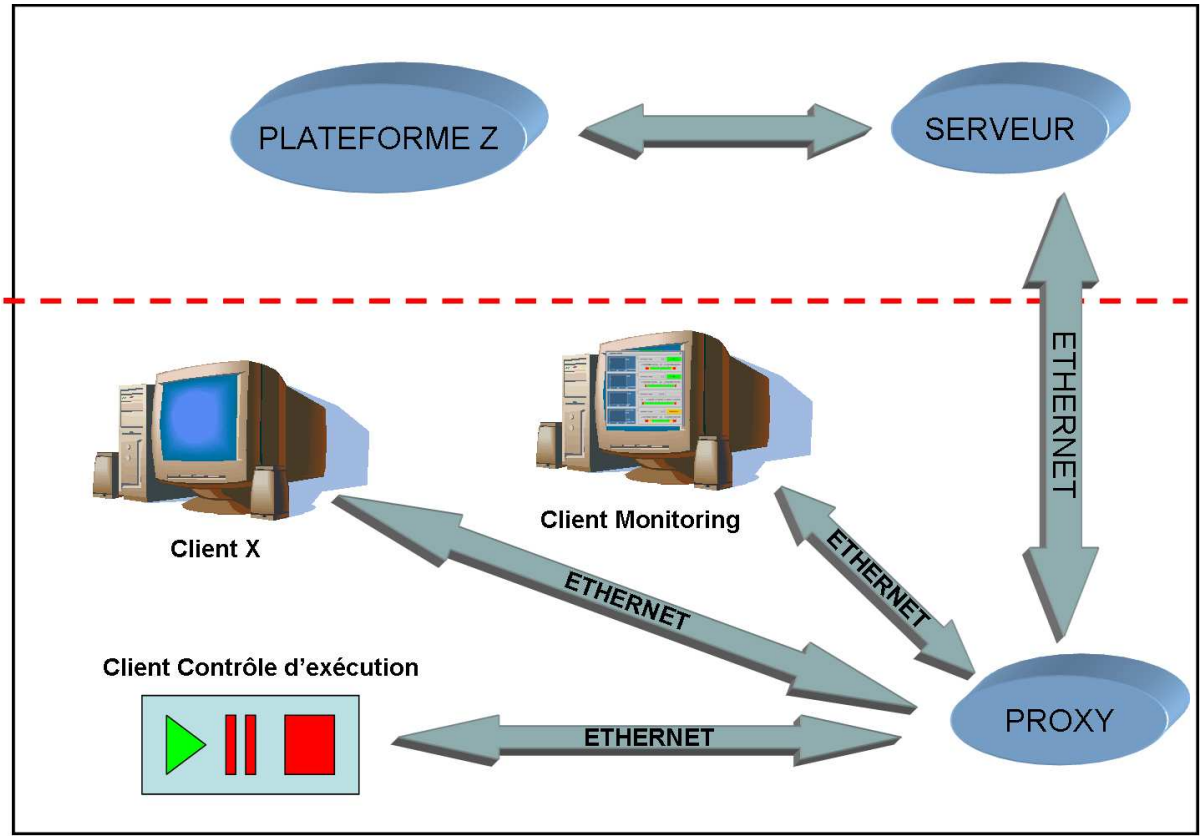

FIG. 6.2 – Architecture globale

Les plates-formes ont des ressources cpu limitées, et sont soumises a des contraintes de temps fortes, elles ne peuvent donc pas directement se charger de la retransmission des données vers l'extérieur sans mettre en danger leur fonctionnement. C'est pourquoi il est primordial d'utiliser un serveur qui joue le rôle de "cache". Les serveurs sont des parties logicielles qui permetent le dialogue entre les plates-formes et l'extérieur. Ils sont soit intégré dans le logiciel qui gère la plate-forme soit déportée dans une machine dédiée. Ils réduisent les accès directs aux plates-formes et améliore les performances globales. Grâce aux serveurs, le monde extérieur ne perturbent pas le fonctionnement du système car il n'accède pas directement aux plates-formes. Je ne rentrerai pas plus dans les détails du fonctionnement des serveurs car

leurs réalisations n'étaient pas le sujet de mon stage.

Le monde extérieur est représenté par les clients. Les clients peuvent avoir un spectre de fonctionnalités variés. On peut regrouper un grand nombre de fonctions dans un seul client qui va assurer la supervision du système ou éclater les fonctionnalités sur plusieurs clients. (ex : la visualisation 3D, l'affichage des données, le contrôle de l'exécution ...) Dans tout les cas le client permet d'interfacer l'utilisateur avec les plates-formes robotique. Aussi on peut donc adapater le besoin de chaque utilisateur de la plate-forme sans avoir a faire de modification du coté serveur. C'est ce qui fait la force de l'architecture "client-serveur". Pour ma part, J'ai choisi de répondre aux spécifications du superviseur sous la forme de plusieurs clients assurant chacun une unique fonctionnalité. Ainsi chaque client assure une tâche bien spécifique. Comme les manipulations sur les plates-formes ne nécessitent pas toujours l'ensemble des fonctionnalités, l'utilisateur pourra choisir d'utiliser tel ou tel client. Il trouve donc une réponse modulaire à son besoin sous la forme de petites applications clientes autonomes. Cette vision apporte une modularité supérieure à l'utilisation d'un unique client qui répondrait à toutes les spécifications du superviseur. La création de client étant spécifique à chaque plate-forme elle ne sera pas traité dans ce chapitre. Des exemples de clients pour la plate-forme BIP sont donnés dans la partie 9.4.

Si cette architecture est assez commune, elle présente tout de même une spécificité, l'utilisation d'un proxy. Il assure l'interconnexion du monde extérieur avec les serveurs. Cette mesure améliore la sécurité du système. Elle permet de mieux isoler la partie serveurs et plates-formes de l'extérieur. Le proxy doit assurer quatre fonctions :

- Gérer les connexions-déconnexions des clients.
- Gérer les connexions-déconnexions des serveurs.
- Gérer les ordres ou les données envoyées par les clients pour un serveur.
- Trier puis rediriger les données en provenance d'un serveur vers les clients concernés.

Il permet d'optimiser la bande passante du coté serveur en se chargeant par exemple de la large diffusion d'un même message vers de nombreux clients.
Pour pouvoir concevoir mon propre proxy, j'ai réalisé un petit état de l'art (Annexe A). Il m'a permit de bien comprendre les mécanismes à mettre en oeuvre.
La suite de ce chapitre décrit donc en détail la conception de mon proxy.

Chapitre 7

Conception du proxy

7.1 Architecture du proxy

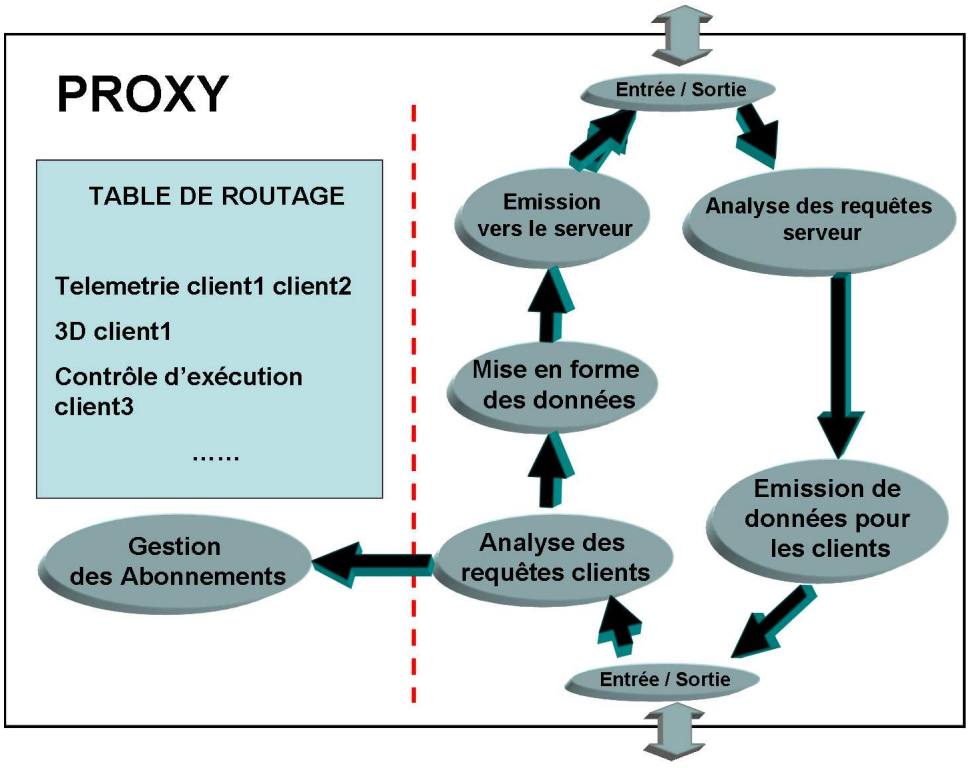

FIG. 7.1 – Architecture du proxy

7.2 Principe de fonctionnement

Le proxy est un système transparent chargé du routage des données entre les clients et les serveurs. Il doit pouvoir redistribuer des données générées par les plates-formes vers les clients (monitoring des données), et il doit faire suivre les données ou les commandes des clients vers les plates-formes (contrôle d'exécution). Si on prend le point de vue d'un proxy, son unique soucis est de savoir faire suivre les différents messages qu'il reçoit aux bons destinataires. Leurs contenus lui importent peu, il doit uniquement savoir à qui s'adresse tel ou tel message. Pour cela, il suffit de tenir à jour une table de routage de tous les messages possibles avec une correspondance vers les destinataires désirant les recevoir. La clef d'entrée de cette table de routage est donc un mot clef correspondant au type de message.

7.2.1 Protocole de communication

Pour pouvoir dialoguer avec les clients, les plates-formes et le proxy doivent partager un langage ou protocole de communication. J'ai donc défini un protocole simple basé sur des chaînes de caractères, les messages les plus importants étant acquittés. Tous les messages sont identifiés par un type. Ils ont la structure suivante

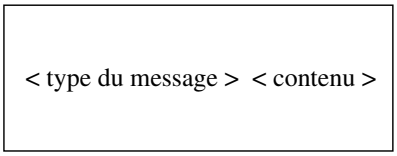

FIG. 7.2 – Format des messages

En fonction du type de message, le contenu $<$ *contenu* $>$ est redivisable. Lors de la mise en place d'une plate-forme sur le proxy on doit lui donner les identifiants des messages. Les sous-parties suivantes présentent les différents dialogues entre les clients, les plates-formes et le proxy.

7.2.2 Gestion des abonnements

La gestion des abonnements est une partie importante du travail du proxy. Elle sert à renseigner la table de routage des destinataires. Quand un client se connecte sur le proxy, il doit choisir les informations qu'il veut recevoir. Pour cela, il s'abonne aux différents messages correspondants. Il demande directement à s'abonner aux messages s'il connaît leurs types, sinon il demande d'abord au proxy la liste des messages que peuvent générer les plates-formes. Le proxy dispose de cette information car elle lui est fournie lors de sa configuration.
Voici donc les messages utilisés pour la gestion des abonnements :

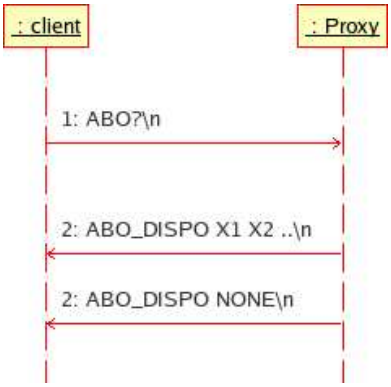

FIG. 7.3 – Echange lors d'une demande des abonnements disponibles

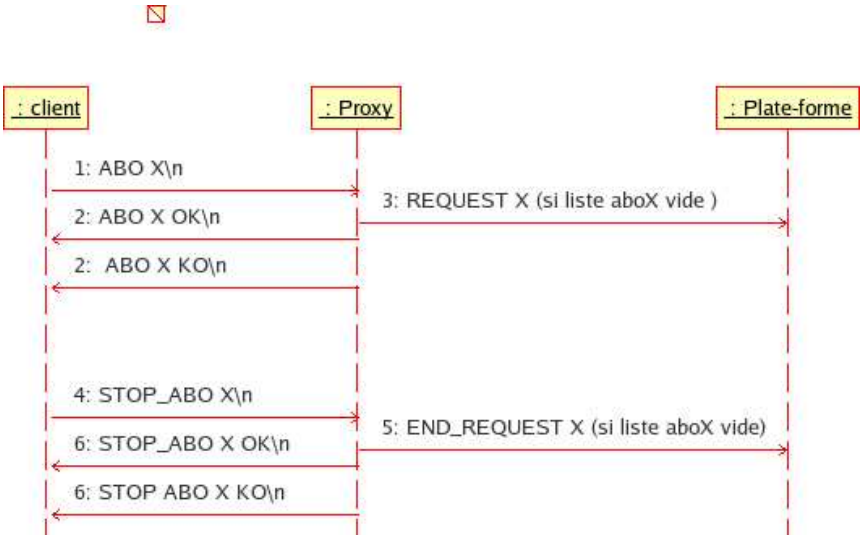

FIG. 7.4 – Echange lors d'un abonnement-désabonnement d'un client

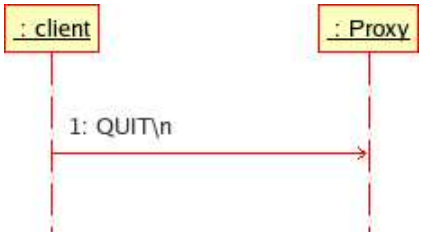

FIG. 7.5 – Echange lors d'une déconnexion d'un client

7.2.3 Monitoring des données

Dans le cas des plates-formes robotiques de l'INRIA, la plupart des messages sont des données de supervisions en provenance des plates-formes. Dès que le proxy reçoit un de ces messages, il cherche dans sa table de routage l'entrée correspondant au type du message, il en déduit les destinataires. Le proxy joue ici le rôle de relais, il ne connaît pas le contenu des messages. Ce sont les clients et les plates-formes qui doivent "se mettre d'accord" sur le format de ces messages. C'est ce qui fait la puissance du proxy car il est générique, en effet il peut venir interfacer de nombreuses plates-formes et clients qui utilisent des protocoles différents.

Voici un exemple de messages de type "TM" pour du monitoring de données :

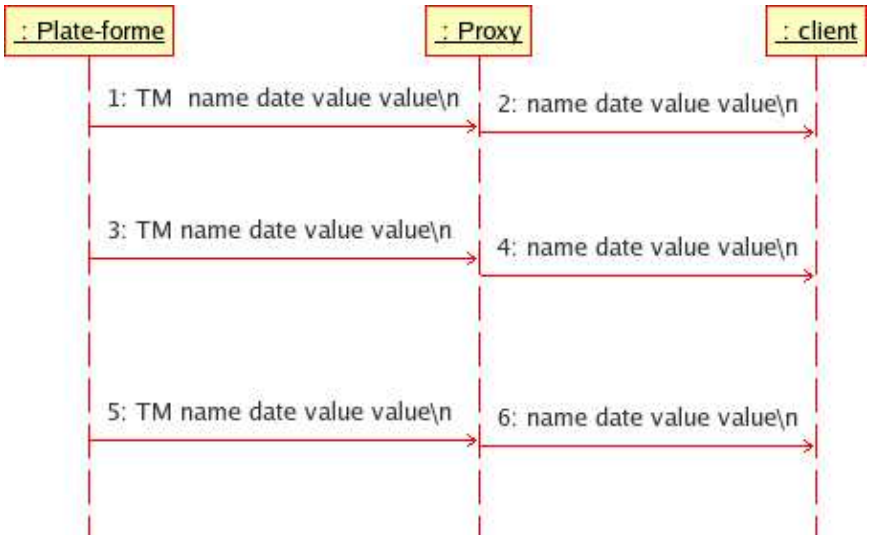

FIG. 7.6 – Echange lors de la réception de données de télémétrie

Ici, on utilise le type de message "TM" mais on peut créer autant de types de messages que l'on veut. De cette manière, on peut par exemple utiliser un seul type de message qui contient toute la télémétrie ou plusieurs types messages pour découper les données de télémétrie. Cette possibilité est très utile car elle apporte beaucoup de flexibilité. En effet, on peut choisir d'envoyer toutes les données en un seul message. Dans ce cas, les clients font le tri des données qui les intéresse. Ou alors on découpe les données de télémétrie en sous-ensembles directement exploitables par les clients. Les clients choisissent alors uniquement les messages qu'ils souhaitent recevoir. On peut donc placer la mise en forme des données soit du coté client soit du coté plate-forme.

7.2.4 Gestion du contrôle d'exécution

Pour le contrôle d'exécution, le proxy doit être capable de faire suivre des données d'un client vers une plate-forme. De plus, pour assurer la sécurité de la plate-forme, le contrôle d'exécution doit se faire de manière atomique. Si plusieurs clients désirent contrôler la plate-forme, ils le feront chacun leur tour. Un client ne peut pas avoir accès au contrôle de la plate-forme si un autre client l'a déjà. Le proxy gère les demandes de contrôle de la plate-forme par une liste d'attente de type "premier arrivé premier servi". Le proxy doit donc être capable de reconnaître les requêtes de contrôle d'exécution. Il utilise donc un type de message spécifique dédié au contrôle d'exécution. Une fois encore, le contenu reste inconnu au proxy, il reconnaît juste le type de message "contrôle d'exécution". A la réception d'un de ces messages il vérifie si le client peut faire du contrôle d'exécution. Si le client est autorisé il fait suivre la requête à la plate-forme. En fonction de la plate-forme utilisée, on peut choisir le mot-clé qui définit le type de message "contrôle d'exécution".

Voici un exemple possible avec le mot-clé "CMD" :

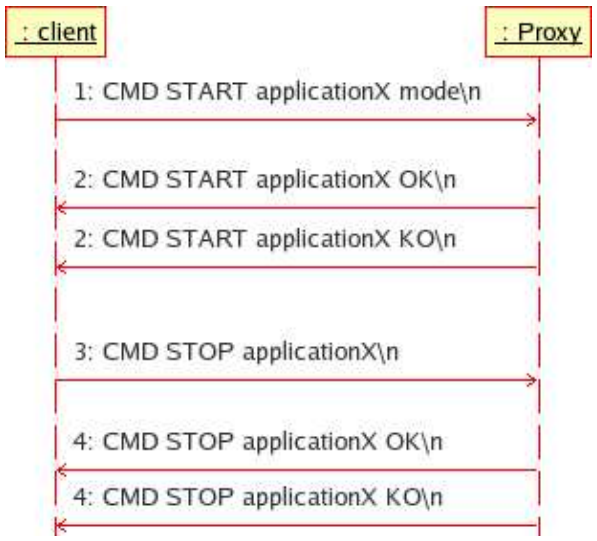

FIG. 7.7 – Echange lors d'une action de contrôle d'exécution

7.3 Réalisation

J'ai choisi de coder cette application en JAVA car la puissance de son API [30] facilite grandement le développement. En effet, les classes de gestion des sockets, des listes chaînées, tables, inhérante au langage répondent parfaitement aux besoins de l'application. D'un point de vue performance, le nombre de clients et de serveurs venant à se connecter sur le proxy est assez limité. Par conséquent, le fait que le JAVA soit un langage interprété moins performant qu'un langage compilé ne posera pas de problème ici. JAVA étant un langage de programmation objet, il favorise la construction d'applications modulaires et réutilisables.

7.3.1 Diagramme de classe

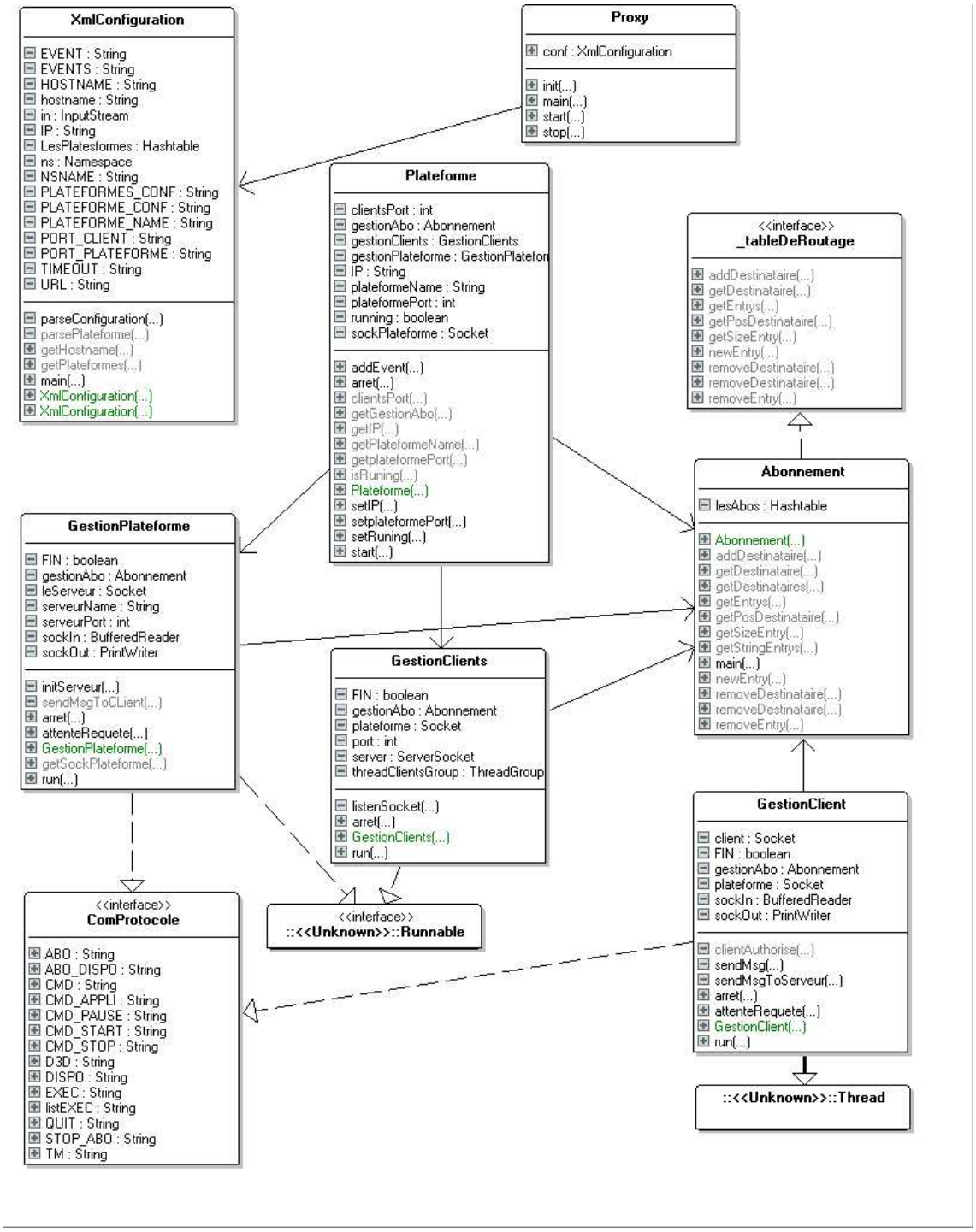

FIG. 7.8 – Diagramme de classe du proxy

la classe Proxy :

Cette classe est le point d'entrée de l'application proxy : elle l'initialise au moyen d'un fichier XML. Ce fichier contient les différentes informations nécessaires à la mise en place des plates-formes sur le proxy. Après l'initialisation, elle lance les plates-formes sur le proxy.

la classe XmlConfiguration :

Cette classe est conçue pour analyser un document XML afin d'obtenir la configuration des différentes plates-formes à gérer sur le proxy.

la classe Plateforme :

La classe Plateforme assure la modélisation d'une plate-forme pour le proxy. Elle contient différentes informations qui lui sont propres. Elle dispose des liens vers un gestionnaire de plate-forme (GestionPlateforme), un gestionnaire de client (GestionClient) et un gestionnaire d'abonnement.

la classe GestionPlateforme :

Cette classe se met en écoute sur le port défini par la configuration de la classe Plateforme. Ensuite elle attend des messages en provenance du serveur et redirige les données aux clients en utilisant le gestionnaire d'abonnement.

la classe GestionClients :

La classe GestionClients crée un "socketserver" sur le port défini par la configuration. Il attend ensuite la connexion des clients sur le port et crée un processus GestionClient pour chaque client.

la classe GestionCLient :

Elle permet de gérer un client, elle écoute et traite les requêtes du client qui arrive sur une socket de communication.

la classe Abonnement :

La classe abonnement est un gestionnaire semblable à une table de routage. Elle contient un nombre variable de listes des types messages que peut recevoir le proxy. Chaque liste concerne un type de message particulier, et contient un lien vers tous les clients désirant le recevoir.

la classe ComProtocole :

La classe définit le langage à utiliser pour le dialogue entre les clients, le proxy et le serveur.

7.4 Utilisation du proxy

7.4.1 Fichier de configuration

Afin de pouvoir paramétrer le proxy sans avoir passer par une phase de compilation du code source, je me base sur un fichier de configuration en XML. Celui-ci peut être modifié manuellement avec un éditeur ou encore avec une application spécialisée. Le fichier contient les informations nécessaires à la mise en place des plates-formes sur le proxy.

Voici la liste exhaustive de ces informations :

- le nom de la plate-forme ;
- l'IP du serveur de la plate-forme ;
- le port du serveur de la plate-forme ;
- le port de connections des clients sur le proxy ;
- la liste des évènements générés par la plate-forme ;

Automate de fonctionnement

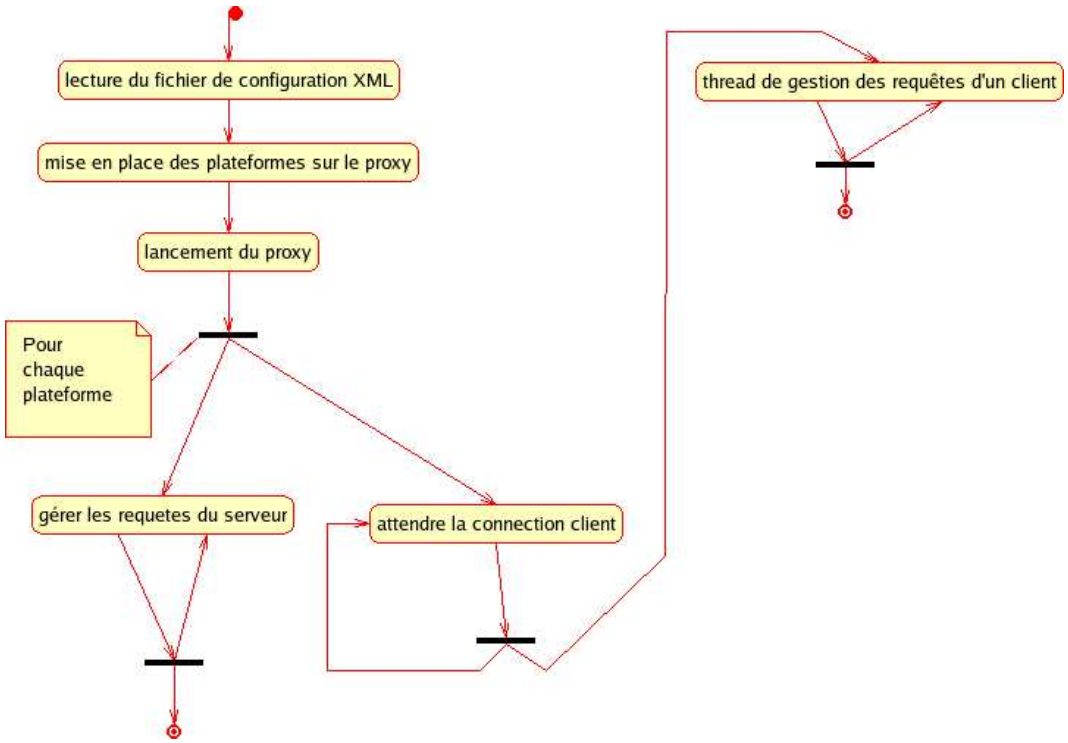

FIG. 7.9 – Automate de fonctionnement du proxy

Après la conception de l'architecture, et réalisation du proxy en JAVA, j'ai validé mon système de supervision sur la plate-forme robotique BIP. La partie suivante présente ce travail.

Troisième partie

Application de la supervision à la plate-forme BIP

Chapitre 8

Présentation de la plate-forme

8.1 Présentation générale

L'objectif initial du projet BIP était la réalisation d'un robot bipède à 17 degrés de liberté (ddl), aux dimensions et masses des membres inférieurs voisines de celles de l'homme. Ce robot a été conçu pour reproduire une marche dynamique 3D stable sur sol plat et assurer la montée et la descente d'escaliers. Il permet l'étude des lois qui caractérisent la marche humaine et il sert de support aux projets BIPOP et POP ART de l'INRIA.

La partie mécanique du robot a été réalisée par le LMS (Laboratoire de Mécanique des Solides) de l'Université de Poitiers. La partie electonique de commande fut réalisé par l'INRIA. Dans sa première version (Juillet 1999) le système ne comportait que deux jambes (8 ddl) associées à un chariot à quatre roues (6 ddl plan). Un tronc intégrant l'électronique de commande a été ajouté au prototype suivant (Janvier 2000). Cette version totalise ainsi 15 ddl dont une rotule complète (3 ddl) au niveau du tronc (cf figure 8.1). Il mesure 180 cm pour 105 kg. Actuellement il existe deux exemplaires de BIP, le premier à l'INRIA Rhône-Alpes et le second au LMS. L'INRIA Rhône-Alpes est chargé de la partie expérimentation et développement logiciel autour du BIP.

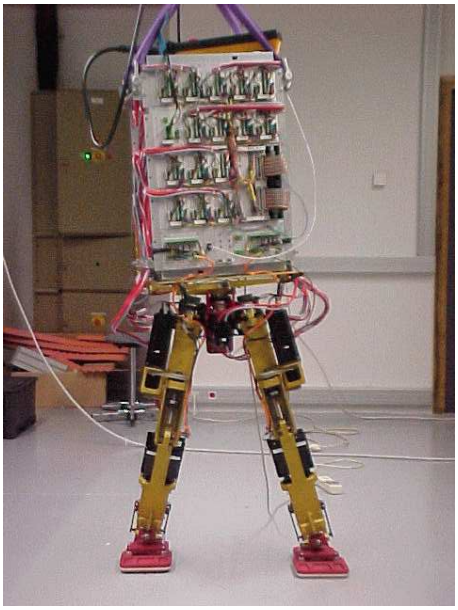

FIG. 8.1 – Le robot bipède BIP et ses degrés de liberté

8.2 Composition du BIP

Le robot BIP se décompose en quatre parties :
- la partie mécanique
- La chaîne électromécanique
- l'armoire de commande [3]
- le contrôleur du robot [2]

8.2.1 La structure mécanique

La structure mécanique du robot bipède est réalisée en aluminium. Elle comprend environ 280 pièces ; elles ont toutes été conçues et en grande partie réalisées par le LMS (Laboratoire de Mécanique des Solides) de l'Université de Poitiers.

8.2.2 La chaîne électromécanique

Les actionneurs

Le robot bipède possède 15 articulations entraînées par 15 moteurs à courant continu sans balai et pilotés par 15 variateurs (SBS).

Les capteurs

Trois capteurs d'effort placés sur chaque pied permettent de mesurer la composante verticale de la force de réaction du sol (force de pression). On utilise des potentiomètres qui permettent de déterminer la position articulaire absolue du robot à sa mise sous tension. On utilise également les informations des codeurs des variateurs pour la position articulaire relative en cours d'exécution. En effet les codeurs sont plus précis que les potentiomètres.

8.2.3 L'armoire de commande

Le but de cette armoire est d'embarquer toute l'électronique de puissance et de commande.

8.2.4 Architecture informatique

Le robot bipède utilise une carte embarquée MVME162 doté d'un CPU MOTOROLA MC68040 [28] piloté par le système d'exploitation VxWorks [19, 18, 29]. Les programmes sont d'abord compilés de façon croisée sous station SUN Solaris puis une commande on télécharge le code sur la carte cible MVME162 via le système VxWorks. Bientôt cette carte sera remplacée par une carte MPC8260 à base de PowerPc 200 MHz plus puissante. La carte actuelle ne dispose pas de la puissance requise pour les calculs de la nouvelle loi qui régit le robot. Actuellement la solution retenue pour palier ce problème est l'utilisation d'un calculateur déporté sur une autre machine. Cette solution sera sûrement conservée même avec la nouvelle carte sous linux car sa puissance sera encore trop limitée. On réservera donc ses ressources pour les procédures d'acquisition des nombreux capteurs. De plus cette dernière, offre une interface réseau 100Mb au lieu des 10Mb actuels, ce qui va résoudre certains problèmes de transfert des données vers la machine déportée.

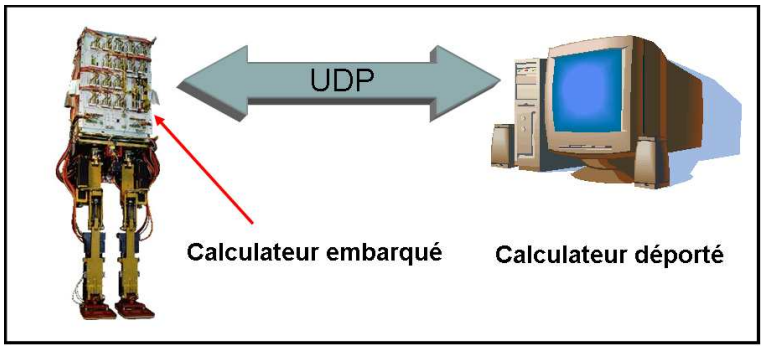

FIG. 8.2 – La plateforme BIP et son calculateur déporté

Chapitre 9

Système de supervision de la plate-forme

9.1 Définition du besoin

Actuellement, les expérimentations sur la plate-forme Bip, et l'exploitation des résultats sont très complexes. Il faut utiliser de nombreux scripts de configuration et des outils non graphiques qui ne sont pas à la portée de tous les utilisateurs de cette plate-forme. De plus, le système ne dispose pas de système de visualisation des données en temps réel. L'INRIA a donc besoin d'un système de supervision capable à la fois de remplacer tous les scripts et outils non graphiques mais aussi d'un système capable de présenter en temps réel les données d'exploitation de la plate-forme. Ce système sera utilisé pour contrôler et analyser le déroulement des expérimentations sur la plate-forme.

Le chapitre suivant établit donc l'adaptation du superviseur présenté dans le chapitre 6 à la plate-forme du bipède.

9.2 Spécifications fonctionnelles

9.2.1 Visualisation des données

Pour la plate-forme du bipède voici les données que l'on souhaite visualiser à travers le superviseur.

Visualisation des données de la plate-forme

Le robot bipède génère lors de son fonctionnement un ensemble d'informations appelés télémétrie.

- **État des capteurs :**
  - Potentiomètres : ils fournissent la position angulaire relative des axes de rotation du bipède.
  - Capteurs de pression : ils expriment les forces normales de pression exercées par le sol sur le bipède.
  - Jauges de contraintes : elles expriment les forces exercées sur la cheville. Cela permet de reconstruire les efforts tangentiels.
  - Fin de courses.
    - matérielles : valeurs angulaires des butées mécaniques des articulations.
    - électriques : valeurs angulaires des butées électriques. Elles expriment les positions angulaires limites à ne pas dépasser sous peine d'arrêt d'urgence.
    - logicielles : valeurs numériques des limites angulaires fixées par l'utilisateur.
  - Inclinomètres : Ils fournissent une représentation des orientations du robot dans l'espace.
- **État des actionneurs :**
  - Position des moteurs.
  - Couples articulaires : couple exercé sur chaque articulation.
  - Positions articulaires.
  - Tensions moteur : elles représentent les tensions effectives lues sur les variateurs.
  - Consignes Moteurs : elles représentent les consignes de tension appliquées aux variateurs.
- **Alarmes :**
  - Arrêt d'urgence : indique que le système a été arrêté électriquement par l'utilisateur.
  - Défaut du driver : défaut sur les encodeurs, module IP, fonctions de haut niveau ...
  - Fin de course logicielle : une position de butée virtuelle définie par l'utilisateur est atteinte.
  - Fin de course matérielle : une position de butée mécanique est atteinte.
  - Saturation des moteurs : les valeurs des commandes moteur sont trop importantes par rapport aux limites fixées.

Visualisation en 3D du robot

Afin de pourvoir vérifier les mouvements du robot lors d'une manipulation de suivi de trajectoire, on souhaite à travers le superviseur représenter les déplacement du robot dans un espace 3D virtuel.

9.2.2 Contrôle de l'exécution

Voici le contrôle d'éxécution désiré pour le bipède :

- Charger une application ;
- Lancer, stopper l'application ;
- Mettre en pause le déroulement de l'application courante ;
- Afficher l'état de l'application : arrêté, en initialisation, en exécution, fin d'exécution ...

9.2.3 Gestion de l'historique

Toutes les données recueillies lors des manipulations du bipède devront être sauvegardées de manière organisée. Les fonctionnalités offertes à l'utilisateur sont :

- Recharger des manipulations dans le superviseur ;
- Rejouer des manipulations dans le superviseur ;
- Sauvegarder des résultats et des manipulations pour les exploiter a posteriori.

9.3 Architecture

L'architecture est celle présentée au paragraphe 6.3 mais adaptée en fonction du bipède comme présenté en figure 9.1.

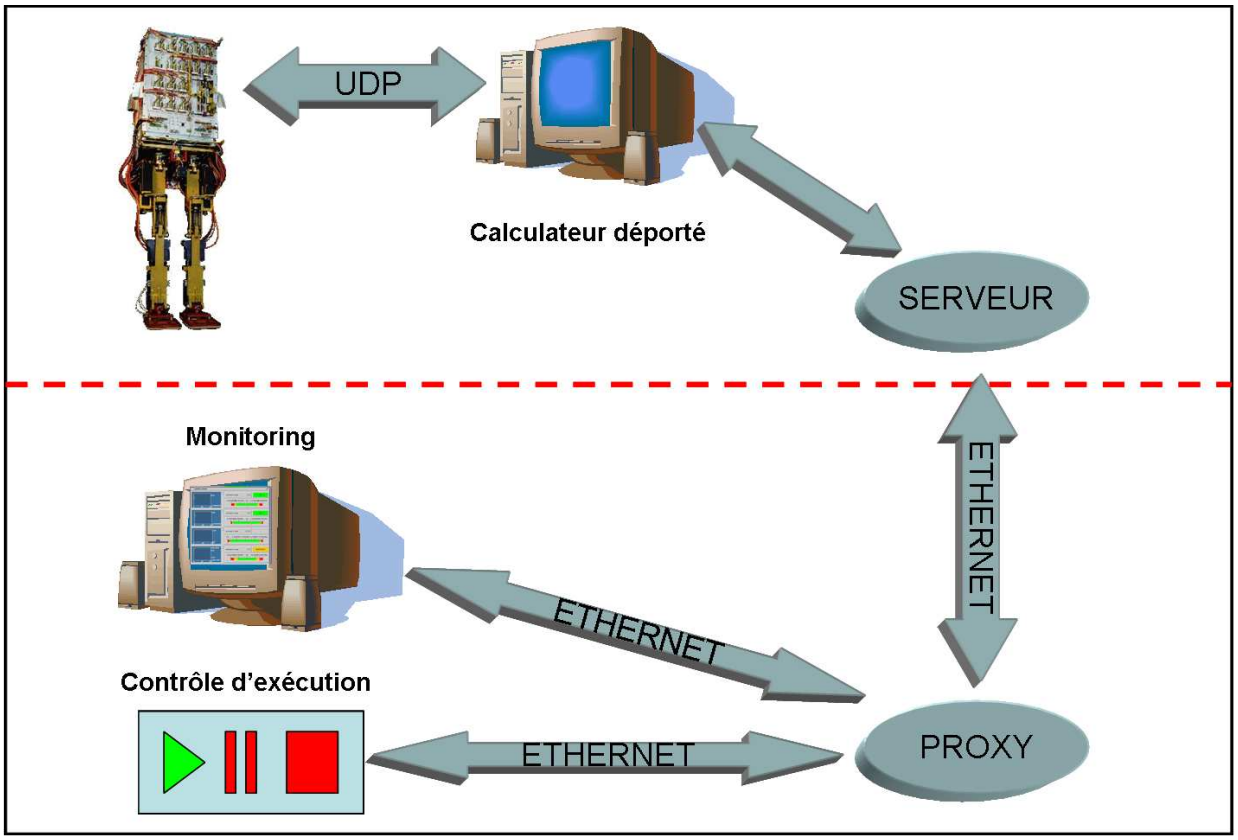

FIG. 9.1 – Architecture générale du superviseur pour la plate-forme BIP

Le proxy ayant déjà été réalisé, pour l'implémentation du superviseur, il ne me restait plus qu'a définir les clients adaptés aux besoins de supervision de la plate-forme BIP. Comme il n'existait pas de contrôleur d'exécution, j'ai conçu et réalisé ce client. Pour le monitoring des données en 2D et en 3D, j'ai réutilisé et adapté des logiciels existants (reenginering).

9.4 Le client contrôle d'exécution

9.4.1 Rôle du client de contrôle d'exécution

Le but de ce client est de pouvoir contrôler à distance le déroulement d'applications sur la plate-forme BIP.

9.4.2 Fonctionnalités

Le contrôleur d'exécution doit assurer cinq fonctions primaires :

- Se connecter sur la plate-forme via le proxy ;
- Choisir puis charger une application sur la plate-forme ;
- Lancer, arrêter l'application en cours d'exécution ;
- Stopper momentanément l'application en cours d'exécution ;
- Avertir l'utilisateur des éventuels problèmes rencontrés sur la plate-forme.

9.4.3 Architecture du client de contrôle d'exécution

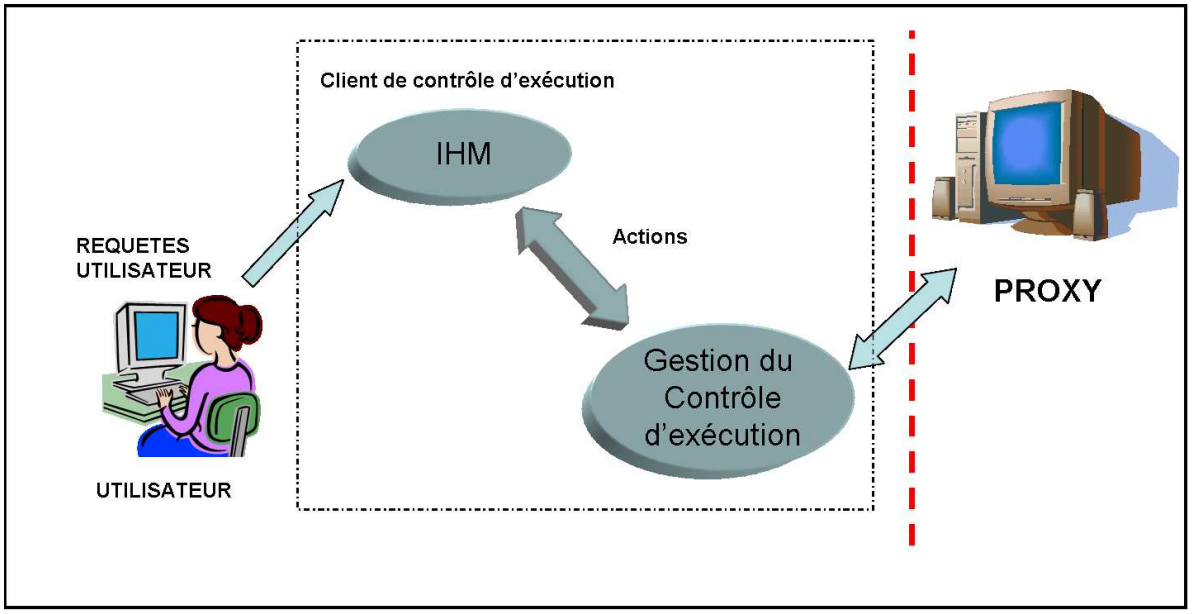

FIG. 9.2 – Architecture du client de contrôle d'exécution

9.4.4 Réalisation

J'ai choisi d'utiliser JAVA pour ce client afin de pourvoir le lancer sur n'importe quelle type de machine indépendamment de l'OS.

Diagramme de classe

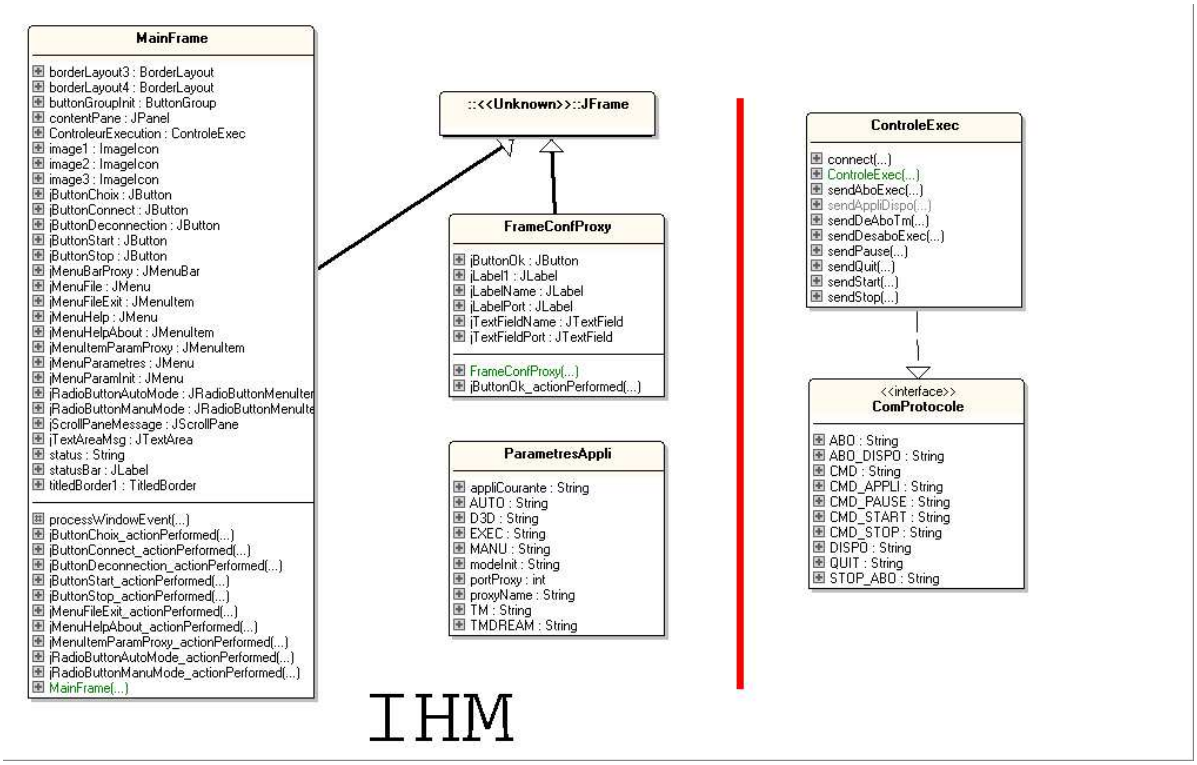

FIG. 9.3 – Diagramme de classe de l'IHM du client contrôle d'exécution

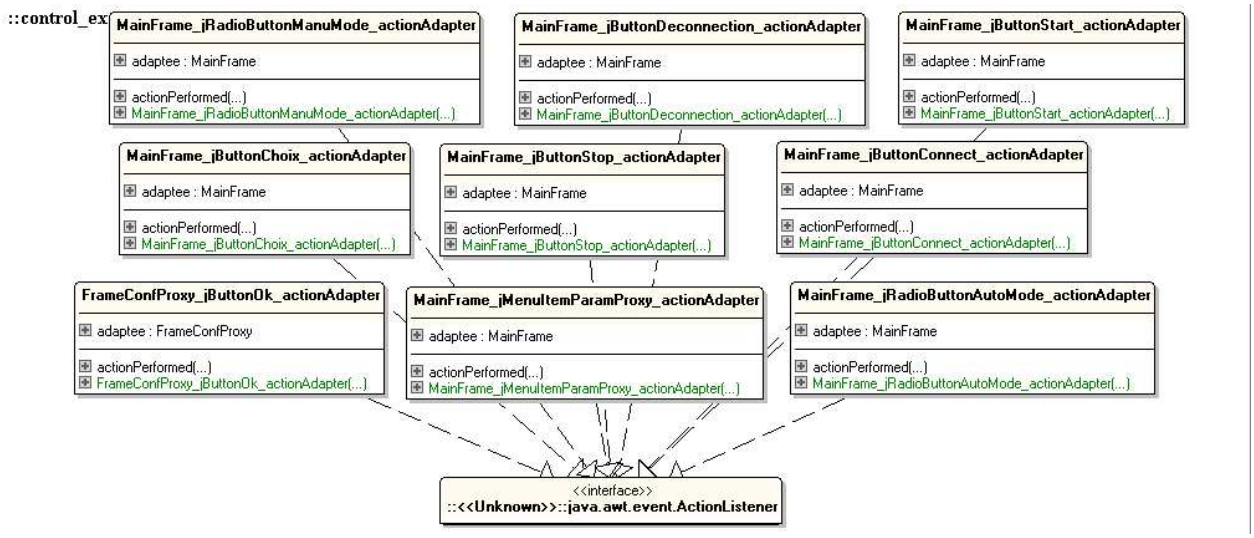

FIG. 9.4 – Diagramme de classe de la gestion des events du client contrôle d'exécution

Et voila une illustration l'IHM du contrôleur d'écécution (cf ANNEXE C)

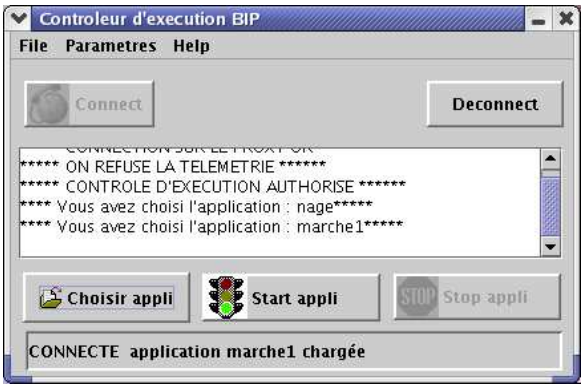

FIG. 9.5 – IHM du client de contrôle d'exécution

9.5 Le client de monitoring des données

9.5.1 Rôle du client de monitoring des données

Le but de ce client est de pouvoir contrôler à distance l'état de la plate-forme et des différentes informations qu'elle génère lors de son exploitation. Ce client correspond à une interface graphique fournissant à l'utilisateur une représentation graphique de la télémétrie de la plate-forme.

9.5.2 Fonctionnalités

Le client de monitoring des données doit assurer les fonctions suivantes :

- Se connecter sur la plate-forme via le proxy ;
- Récupérer les données de télémétrie ;
- Visualiser la télémétrie ;
- Visualiser les alarmes ;
- Sauvegarder la télémétrie et les alarmes d'une manipulation ;
- Rejouer la télémétrie et les alarmes d'une manipulation ;

9.5.3 Architecture du client de monitoring des données

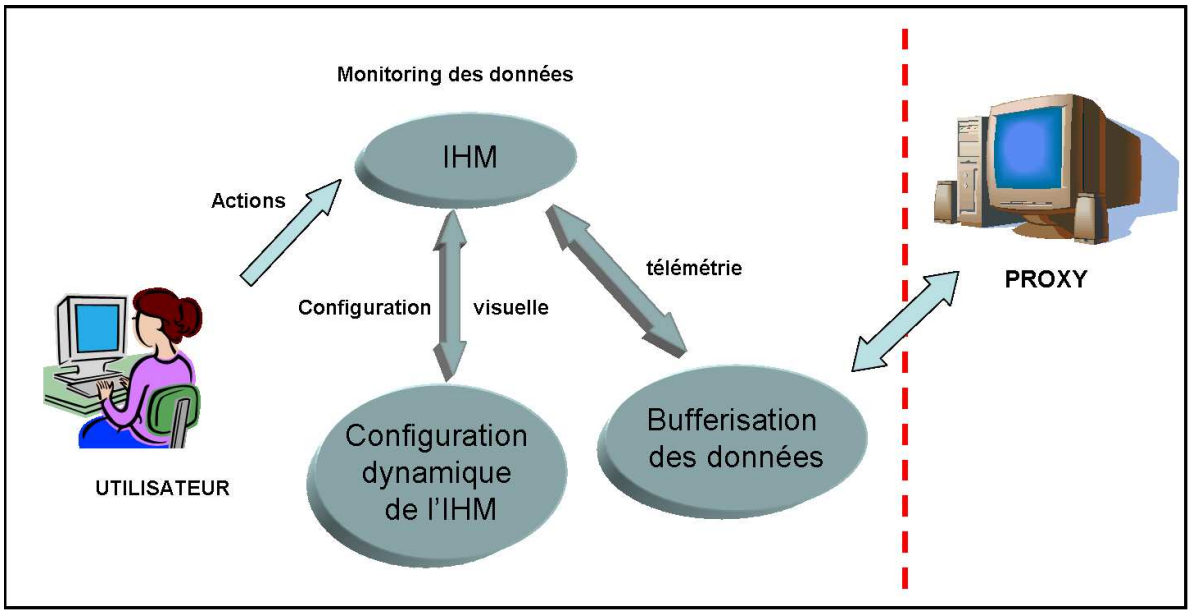

FIG. 9.6 – Architecture du client de monitoring des données

9.5.4 DREAM "light"

Comme déjà dit au chapitre 5, le service SED avait à sa disposition le logiciel DREAM. Comme déjà dit ce logiciel offre une interface puissante pour assurer le . J'ai donc pu la réutiliser. Grâce cette réutilisation j'ai gagné 5 mois de programmation ce qui m'a permis d'orienter ma réflexion sur la partie proxy. J'ai donc fourni à TRASYS un compte-rendu des problèmes rencontrés lors de l'adaptation de leur logiciel à nos besoins. Ce compte-rendu nous a permis d'obtenir une version "allégé" de DREAM " DREAM light" et permet à la société TRASYS d'avoir un retour client lui permettant de corriger quelques défauts.

Par abus de langage, j'utiliserai dans la suite de cette section le terme DREAM pour la partie monitoring des données du logiciel. La configuration du logiciel DREAM se fait grâce à des fichiers XML. Un des fichiers permet de définir les données de télémétrie à recevoir et le deuxième leur représentation.

Configuration du fichier telemetry.xml

Pour chaque valeur de capteur ou autre information générée par la plate-forme BIP, j'ai défini une entrée dans le fichier de configuration. Voici, par exemple, l'entrée du fichier XML relative à la cheville droite dans le plan frontal.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Telemetries SYSTEM "Telemetry.dtd">
<!-- on crée une télémétrie  -->
<Telemetries>
    <!-- elle est de type Real  -->
    <Telemetrie TmType="Real">
    <!-- son nom est Bipjointure -->
    <TmName>Bipjointure</TmName>
    <!-- les arguments sont reçu en permanence -->
    <Argmnt ArgType="Continuous">
      <!-- on crée un argument ArtCheD_F  -->
      <ArgName>ArtCheD_F</ArgName>
      <!-- l'unité est le radian   -->
      <ArgUnit>rad</ArgUnit>
      <!-- la valeur minimale prise par l'argument est   -->
      <Ymin>-0.3490658504</Ymin>
      <!-- la valeur maximale prise par l'argument est   -->
      <Ymax>0.349065850</Ymax>
      <!-- on enregistre les valeurs dans le fichier de log   -->
      <Logged>true</Logged>
      <!-- on définit des alarmes sur l'argument   -->
      <Alerte>
```

```
      <!-- zone de fonctionnement normal   -->
<AlName>Ok</AlName>
      <AlType>OK</AlType>
      <LowValue>-0.24</LowValue>
      <HighValue>0.24</HighValue>
     </Alerte>
      <Alerte>
      <!-- zone de dangereuse   -->
<AlName>Warning</AlName>
      <AlType>WARNING</AlType>
      <LowValue>-0.2792527</LowValue>
      <HighValue>-0.24</HighValue>
     </Alerte>
   </Argmnt>
```

La difficulté de ce travail résidait dans le fait que je ne disposais d'aucune documentation. Il a donc fallu que je fasse de nombreux tests pour comprendre le rôle de chaque mot-clé.

Configuration du fichier display.xml

Ce fichier permet de définir la présentation de la télémétrie dans le logiciel DREAM. Il permet de plus de choisir les données à afficher et leurs formes de représentation : graphique, jauges, valeur numérique ... J'ai donc configuré le fichier de manière à disposer d'une représentation des données organisées de la manière suivante :

- jambe droite (positions angulaires)
- jambe gauche (positions angulaires)
- tronc (positions angulaires)
- potentiomètres
- couples moteurs
- tensions moteurs
- alarmes

Voici le résultat visuel :

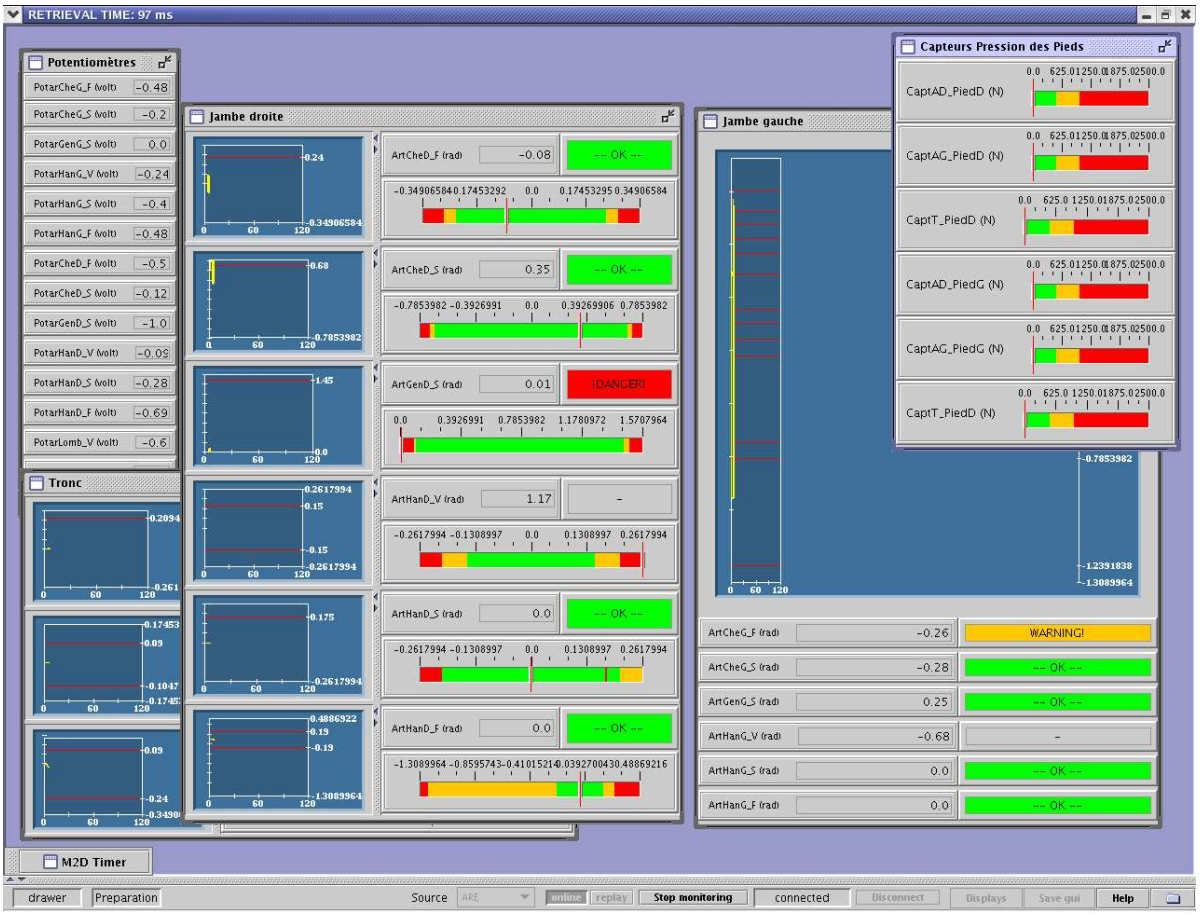

FIG. 9.7 – IHM de DREAM pour la partie "monitoring des données"

Correction des défauts

L'IHM illustrée en figure 9.7, l'IHM présente certains défauts :

- **Gestion des axes des ordonnées**
  L'axe des ordonnées (le temps) est exprimé en minute. La durée de représentation est fixée à 120 minutes et ne peut pas être changée. Cela pose problème car les manipulations sur le bipède ne dépasse jamais quelques minutes. L'échelle est donc inadaptée.
- **Gestion des couleurs**
  DREAM offre comme on peut le voir sur la fenêtre "Jambe Gauche" une fonctionnalité intéressante. Pour gagner de la place il est possible de superposer plusieurs courbes sur

le même graphique. Mais on observe que toutes les courbes sont de la même couleur. Il est donc impossible d'identifier qui est qui. Il semblerait plus intéressant d'utiliser plusieurs couleurs et rajouter des labels pour identifier les courbes. On observe le même problème avec les limites de fonctionnement(traits horizontaux rouges)

Ces défauts qui devrait être rapidement corrigés par TRASYS.

9.6 Le visualisateur 3D

9.6.1 Rôle du visualisateur

Le but de ce client est de pouvoir visualiser les mouvements du robot dans un espace à trois dimensions. Ce client permet d'avoir un aperçu du mouvement réel du robot, il est très utile si l'on n'a pas d'accès visuel au robot.

9.6.2 Fonctionnalités

Le visualisateur 3D doit assurer les fonctions suivantes :
- Se connecter sur la plate-forme via le proxy ;
- Récupérer les données 3D ;
- Visualiser le déplacement du Robot en 3D.

9.6.3 Fonctionnement

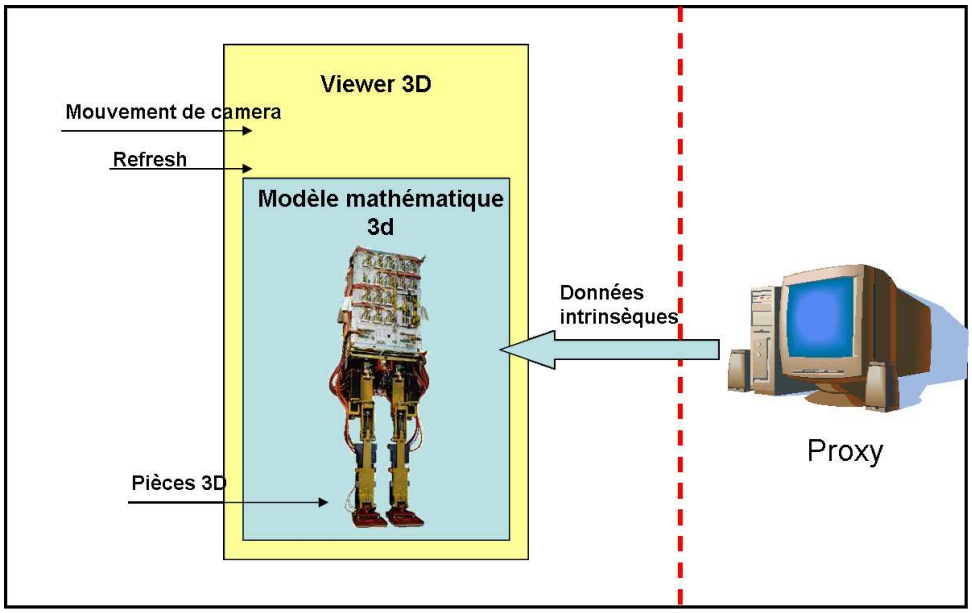

FIG. 9.8 – Architecture du visualisateur 3D

A la base visualisateur 3D a été développé par Pierre Brice Wieber et adapté par Matthieu Guilbert [6] de l'INRIA. Le visualisateur 3D se compose de 2 éléments principaux, le modèle 3D et le viewer 3D. Le modèle 3D est basé est un modèle mathématique qui modélise les positions des pièces du robot sous la forme d'équations. Le viewer 3D QGL Viewer développé

par Gilles Debunne INRIA se charge de la représentation 3D du modèle théorique. Il donne à l'écran une représentation très proche de la réalité grâce à un modèle des pièces du robot en 3D. A chaque itération les variables des équations sont mises a jour grâce aux informations en provenance du proxy. On demande ensuite au viewer 3D de mettre a jour sa représentation ce qui donne le mouvement du robot.

9.6.4 Modification

Le visualisateur 3D a été conçu pour visualiser des trajectoires pré-calculées, enregistrées un fichier de trajectoire.Pour pouvoir l'interfacer avec le système de supervision, j'ai apporté quelques modifications tel que le remplacement du fichier de trajectoire par un flux de données en provenance du Proxy. En effet le visualisateur utilisait un fichier pour obtenir les valeurs des variables des équations. Cette méthode n'était pas utilisable pour une visualisation en temps réel des mouvements. Il est plus simple de se mettre en écoute sur un socket et de mettre à jour le modèle dès la réception de nouvelles valeurs. Cependant il restait encore un problème, la plate-forme utilise un vecteur "positions articulaires" Q de dimension 15 alors que le programme attend un vecteur de dimension 21. Il a donc fallu créer une interface qui recalcule les 6 valeurs manquantes. Ces 6 valeurs correspondent à la position et à l'orientation du robot dans l'espace.

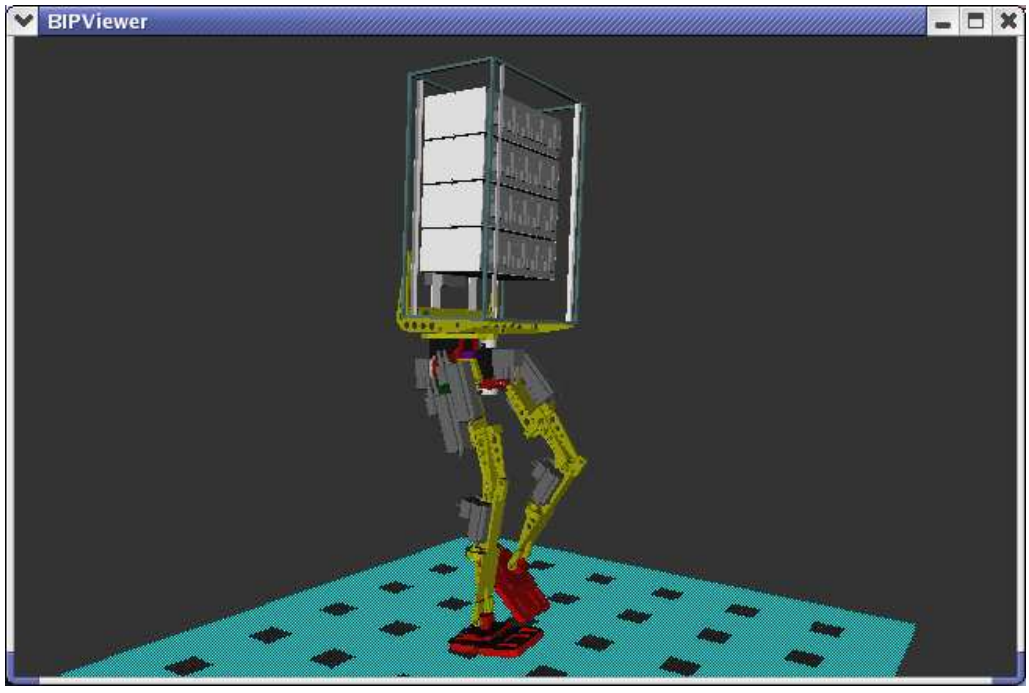

FIG. 9.9 – IHM du visualisateur 3D

Perspectives & Conclusions

L'objectif de ce stage était en premier lieu d'implémenter un système de supervision pour la plate-forme BIP. Finalement, nous avons élargi sa portée à l'ensemble des plates-formes du service. Les ressemblances et les spécificités de chacune m'ont conduit à un concevoir un superviseur générique.

Lors de la première partie de mon stage, j'ai étudié les systèmes de supervision existants. Cette analyse m'a permi de dégager les avantages et les inconvénients de chaque solution et ainsi de retenir la plus appropriée a nos besoins.

Je me suis ensuite orienté vers une partie plus pratique. J'ai alterné avec plaisir développement et reeingineering. Pour ce faire, j'étais en contact avec TRASYS et j'ai pris conscience de l'inertie du travail collaboratif. N'ayant pas de véritable contrat commercial avec cette société, les retours que nous avons fait n'ont pas été encore pris en compte. Dans sa configuration actuel le monitoring sous DREAM "light" n'est pas exploitable. J'ai aussi réalisé l'importance de la conception d'applications modulaires et réutilisables. Elles apportent une souplesse très appréciable lorsque que l'on souhaite apporter des modifications. La mise en place du superviseur sur la plate-forme BIP ne fut pas sans difficultés. Elle subit des modifications constantes et la mise en place de l'architecture intégrant le serveur n'était pas encore au goût du jour. C'est pourquoi il m'était difficile de concevoir des clients pour une architecture sous-jacente encore inexistante. J'ai tout de même pu surmonter ces difficultés avec l'aide des membres du service qui avaient déjà une idée des futurs possibilités du serveur. Aussi, nous avons établi ensemble le futur protocole de communication notamment pour le contrôle d'exécution.

La prochaine étape consiste donc à réaliser le lien entre la partie "clients-proxy" et le serveur, puis d'effectuer les tests d'intégrations final.

Sur un plan plus personnel, ce stage d'une durée de six mois, au sein de l'équipe du service SED m'a permis de participer à un projet de grande envergure et de découvrir le monde de la recherche. Un tel projet exige la collaboration de personnes spécialisées dans des domaines aussi différents que la mécanique, l'électronique, l'informatique temps réel et l'automatique. Le déroulement d'un stage dans un environnement si hétérogène constitue une expérience unique tant sur le point professionnel qu'humain. L'INRIA accueille des personnes aux idées et aux cultures très différentes qui chaque jour apportent leurs lots de nouvelles rencontres et

de nouvelles expériences. Le monde de la recherche reste un milieu en perpétuel effervescence. Je retiendrais cette chaleur humaine et cette envie de partage de la connaissance qui lui est propre. Je n'oublie pas non la complexité des systèmes utilisés et les nombreuse heures de lecture et d'essais pour comprendre leur fonctionnement.

GLOSSAIRE

BIP Robot Bipède de l'INRIA.

CNRS Centre National pour la Recherche Scientifique.

EPST Etablissement public à caractère Scientifique et Technique

ESA Agence Spatiale Européenne.

IHM Interface Homme-Machine.

INRIA Institut National de Recherche en Informatique et Automatique.

LMS Laboratoire de Mécanique des Solides de Poitier.

OS Système d'Exploitation.

SED Service Support Expérimentation et Développement.

UML Unified Modeling Language.

XML Extensible Markup Language.

Quatrième partie

Annexes

Annexe A

le proxy

ETAT DE L'ART

.

CONFIGURATION DU PROXY

fichier *ConfProxy.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<JavaXML:Proxy-config xmlns:JavaXML="http://www.oreilly.com/catalog/javaxml/">

<!-- Données générales sur le proxy -->

<JavaXML:version>1.0</JavaXML:version>
<JavaXML:hostname>AKILA</JavaXML:hostname>
        <!-- Données de configuration des plateformes -->
        <JavaXML:plateformesConfig>
                <!-- Données de configuration de la plateforme bip -->

                <JavaXML:plateformeConfig>
                        <JavaXML:identifier>BIP</JavaXML:identifier>
                        <JavaXML:portClient>7032</JavaXML:portClient>
                        <JavaXML:IPPlateforme>194.199.21.187</JavaXML:IPPlateforme>
                        <JavaXML:portPlateforme>7031</JavaXML:portPlateforme>
                        <JavaXML:timeout>10</JavaXML:timeout>
                        <JavaXML:requestlist>
                                <JavaXML:request_data>TM</JavaXML:request_data>
                                <JavaXML:request_data>DREAM</JavaXML:request_data>
                                <JavaXML:request_ctrl>CMD</JavaXML:request_ctrl>

                        </JavaXML:requestlist>

                </JavaXML:plateformeConfig>
                <!-- Données de configuration de la plateforme RX90 -->
                <JavaXML:plateformeConfig>
                        <JavaXML:identifier>RX90</JavaXML:identifier>
                        <JavaXML:portClient>5006</JavaXML:portClient>
                        <JavaXML:IPPlateforme>194.199.21.187</JavaXML:IPPlateforme>
                        <JavaXML:portPlateforme>5000</JavaXML:portPlateforme>
                        <JavaXML:timeout>10</JavaXML:timeout>
                        <JavaXML:requestlist>
                                <JavaXML:request_data>TM</JavaXML:request_data>
                                <JavaXML:request_data>DREAM</JavaXML:request_data>
                                <JavaXML:request_ctrl>CMD</JavaXML:request_ctrl>
                        </JavaXML:requestlist>
                </JavaXML:plateformeConfig>
        </JavaXML:plateformesConfig>
</JavaXML:Proxy-config>
```

Annexe B

Configuration du client de monitoring

fichier *telemetrie.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DisplayDefs SYSTEM "Displays.dtd">
<DisplayDefs>
<DisplaysDef DispDefName='defaultSys'>
<Display DispType='Graph' m2Disp='true'>
<DispName>Jambe gauche</DispName>
<Regroup>true</Regroup>
<Horizon>false</Horizon>
<Infos>Text and Alarm and Progress</Infos>
<PosX>15</PosX>
<PosY>41</PosY>
<DimW>503</DimW>
<DimH>775</DimH>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtCheG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtCheG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtGenG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtHanG_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtHanG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtHanG_F</ArgName>
</TelemetrieArg>
</Display>
<Display DispType='Graph' m2Disp='true'>
<DispName>Jambe droite</DispName>
<Regroup>false</Regroup>
<Horizon>false</Horizon>
<Infos>Text and Alarm and Progress</Infos>
<PosX>15</PosX>
<PosY>41</PosY>
<DimW>503</DimW>
<DimH>775</DimH>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtCheD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtCheD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtGenD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtHanD_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtHanD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtHanD_F</ArgName>
</TelemetrieArg>
</Display>
<Display DispType='Graph' m2Disp='true'>
<DispName>Tronc</DispName>
<Regroup>false</Regroup>
<Horizon>false</Horizon>
<Infos>Text and Alarm and Progress</Infos>
<PosX>15</PosX>
<PosY>41</PosY>
<DimW>600</DimW>
<DimH>400</DimH>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtLomb_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtLomb_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtLomb_S</ArgName>
</TelemetrieArg>
</Display>
<Display DispType='Text' m2Disp='true'>
<DispName>Potentiomètres</DispName>
<Regroup>true</Regroup>
<Horizon>false</Horizon>
<Infos>Text</Infos>
<PosX>15</PosX>
<PosY>41</PosY>
<DimW>170</DimW>
<DimH>450</DimH>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarCheG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarCheG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarGenG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarHanG_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarHanG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
```

```
<TmName>BipPotars</TmName>
<ArgName>PotarHanG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarCheD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarCheD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarGenD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarHanD_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarHanD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarHanD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarLomb_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarLomb_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarLomb_S</ArgName>
</TelemetrieArg>
</Display>
<Display DispType='Text' m2Disp='true'>
<DispName>Tensions Moteurs </DispName>
<Regroup>true</Regroup>
<Horizon>false</Horizon>
<Infos>Text</Infos>
<PosX>15</PosX>
<PosY>50</PosY>
<DimW>170</DimW>
<DimH>450</DimH>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensCheG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensCheG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensGenG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensHanG_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensHanG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensHanG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensCheD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensCheD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensGenD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensHanD_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensHanD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensHanD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensLomb_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensLomb_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensLomb_S</ArgName>
</TelemetrieArg>
</Display>
<Display DispType='Text' m2Disp='true'>
<DispName>Consignes Moteurs </DispName>
<Regroup>true</Regroup>
<Horizon>false</Horizon>
<Infos>Text</Infos>
<PosX>15</PosX>
<PosY>80</PosY>
<DimW>170</DimW>
<DimH>450</DimH>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsCheG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
```

```
<TmName>ConsMoteurs</TmName>
<ArgName>ConsCheG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsGenG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsHanG_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsHanG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsHanG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsCheD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsCheD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsGenD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsHanD_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsHanD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsHanD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsLomb_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsLomb_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsLomb_S</ArgName>
</TelemetrieArg>
</Display>


<Display DispType='Text' m2Disp='true'>
<DispName>Positions Moteurs </DispName>
<Regroup>false</Regroup>
<Horizon>false</Horizon>
<Infos>Text</Infos>
<PosX>15</PosX>
<PosY>70</PosY>
<DimW>170</DimW>
<DimH>450</DimH>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotCheG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotCheG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotGenG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotHanG_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotHanG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotHanG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotCheD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotCheD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotGenD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotHanD_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotHanD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotHanD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotLomb_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotLomb_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
```

```
<TmName>PosMoteurs</TmName>
<ArgName>PosMotLomb_S</ArgName>
</TelemetrieArg>
</Display>
<Display DispType='Progress' m2Disp='true'>
<DispName>Capteurs Pression des Pieds</DispName>
<Regroup>true</Regroup>
<Horizon>false</Horizon>
<Infos>Text</Infos>
<PosX>952</PosX>
<PosY>8</PosY>
<DimW>311</DimW>
<DimH>450</DimH>
<TelemetrieArg>
<TmName>CaptPressPieds</TmName>
<ArgName>CaptAD_PiedD</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>CaptPressPieds</TmName>
<ArgName>CaptAG_PiedD</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>CaptPressPieds</TmName>
<ArgName>CaptT_PiedD</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>CaptPressPieds</TmName>
<ArgName>CaptAD_PiedG</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>CaptPressPieds</TmName>
<ArgName>CaptAG_PiedG</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>CaptPressPieds</TmName>
<ArgName>CaptT_PiedD</ArgName>
</TelemetrieArg>
</Display>
</DisplaysDef>
</DisplayDefs>
```

fichier *displays.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DisplayDefs SYSTEM "Displays.dtd">
<DisplayDefs>
<DisplaysDef DispDefName='defaultSys'>
<Display DispType='Graph' m2Disp='true'>
<DispName>Jambe gauche</DispName>
<Regroup>true</Regroup>
<Horizon>false</Horizon>
<Infos>Text and Alarm and Progress</Infos>
<PosX>15</PosX>
<PosY>41</PosY>
<DimW>503</DimW>
<DimH>775</DimH>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtCheG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtCheG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtGenG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtHanG_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtHanG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtHanG_F</ArgName>
</TelemetrieArg>
</Display>
<Display DispType='Graph' m2Disp='true'>
<DispName>Jambe droite</DispName>
<Regroup>false</Regroup>
<Horizon>false</Horizon>
<Infos>Text and Alarm and Progress</Infos>
<PosX>15</PosX>
<PosY>41</PosY>
<DimW>503</DimW>
<DimH>775</DimH>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtCheD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtCheD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtGenD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtHanD_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtHanD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtHanD_F</ArgName>
</TelemetrieArg>
</Display>
<Display DispType='Graph' m2Disp='true'>
<DispName>Tronc</DispName>
<Regroup>false</Regroup>
<Horizon>false</Horizon>
<Infos>Text and Alarm and Progress</Infos>
<PosX>15</PosX>
<PosY>41</PosY>
<DimW>600</DimW>
<DimH>400</DimH>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtLomb_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtLomb_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>Bipjointure</TmName>
<ArgName>ArtLomb_S</ArgName>
</TelemetrieArg>
</Display>
<Display DispType='Text' m2Disp='true'>
<DispName>Potentiomètres</DispName>
<Regroup>true</Regroup>
<Horizon>false</Horizon>
<Infos>Text</Infos>
<PosX>15</PosX>
<PosY>41</PosY>
<DimW>170</DimW>
<DimH>450</DimH>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarCheG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarCheG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarGenG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarHanG_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarHanG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
```

```
<TmName>BipPotars</TmName>
<ArgName>PotarHanG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarCheD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarCheD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarGenD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarHanD_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarHanD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarHanD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarLomb_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarLomb_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>BipPotars</TmName>
<ArgName>PotarLomb_S</ArgName>
</TelemetrieArg>
</Display>
<Display DispType='Text' m2Disp='true'>
<DispName>Tensions Moteurs </DispName>
<Regroup>true</Regroup>
<Horizon>false</Horizon>
<Infos>Text</Infos>
<PosX>15</PosX>
<PosY>50</PosY>
<DimW>170</DimW>
<DimH>450</DimH>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensCheG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensCheG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensGenG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensHanG_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensHanG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensHanG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensCheD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensCheD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensGenD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensHanD_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensHanD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensHanD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensLomb_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensLomb_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>TensMoteurs</TmName>
<ArgName>TensLomb_S</ArgName>
</TelemetrieArg>
</Display>
<Display DispType='Text' m2Disp='true'>
<DispName>Consignes Moteurs </DispName>
<Regroup>true</Regroup>
<Horizon>false</Horizon>
<Infos>Text</Infos>
<PosX>15</PosX>
<PosY>80</PosY>
<DimW>170</DimW>
<DimH>450</DimH>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsCheG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
```

```
<TmName>ConsMoteurs</TmName>
<ArgName>ConsCheG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsGenG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsHanG_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsHanG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsHanG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsCheD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsCheD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsGenD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsHanD_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsHanD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsHanD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsLomb_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsLomb_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>ConsMoteurs</TmName>
<ArgName>ConsLomb_S</ArgName>
</TelemetrieArg>
</Display>


<Display DispType='Text' m2Disp='true'>
<DispName>Positions Moteurs </DispName>
<Regroup>false</Regroup>
<Horizon>false</Horizon>
<Infos>Text</Infos>
<PosX>15</PosX>
<PosY>70</PosY>
<DimW>170</DimW>
<DimH>450</DimH>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotCheG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotCheG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotGenG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotHanG_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotHanG_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotHanG_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotCheD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotCheD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotGenD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotHanD_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotHanD_S</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotHanD_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotLomb_V</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>PosMoteurs</TmName>
<ArgName>PosMotLomb_F</ArgName>
</TelemetrieArg>
<TelemetrieArg>
```

```
<TmName>PosMoteurs</TmName>
<ArgName>PosMotLomb_S</ArgName>
</TelemetrieArg>
</Display>
<Display DispType='Progress' m2Disp='true'>
<DispName>Capteurs Pression des Pieds</DispName>
<Regroup>true</Regroup>
<Horizon>false</Horizon>
<Infos>Text</Infos>
<PosX>952</PosX>
<PosY>8</PosY>
<DimW>311</DimW>
<DimH>450</DimH>
<TelemetrieArg>
<TmName>CaptPressPieds</TmName>
<ArgName>CaptAD_PiedD</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>CaptPressPieds</TmName>
<ArgName>CaptAG_PiedD</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>CaptPressPieds</TmName>
<ArgName>CaptT_PiedD</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>CaptPressPieds</TmName>
<ArgName>CaptAD_PiedG</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>CaptPressPieds</TmName>
<ArgName>CaptAG_PiedG</ArgName>
</TelemetrieArg>
<TelemetrieArg>
<TmName>CaptPressPieds</TmName>
<ArgName>CaptT_PiedD</ArgName>
</TelemetrieArg>
</Display>
</DisplaysDef>
</DisplayDefs>
```

Annexe C

Client contrôleur d'exécution

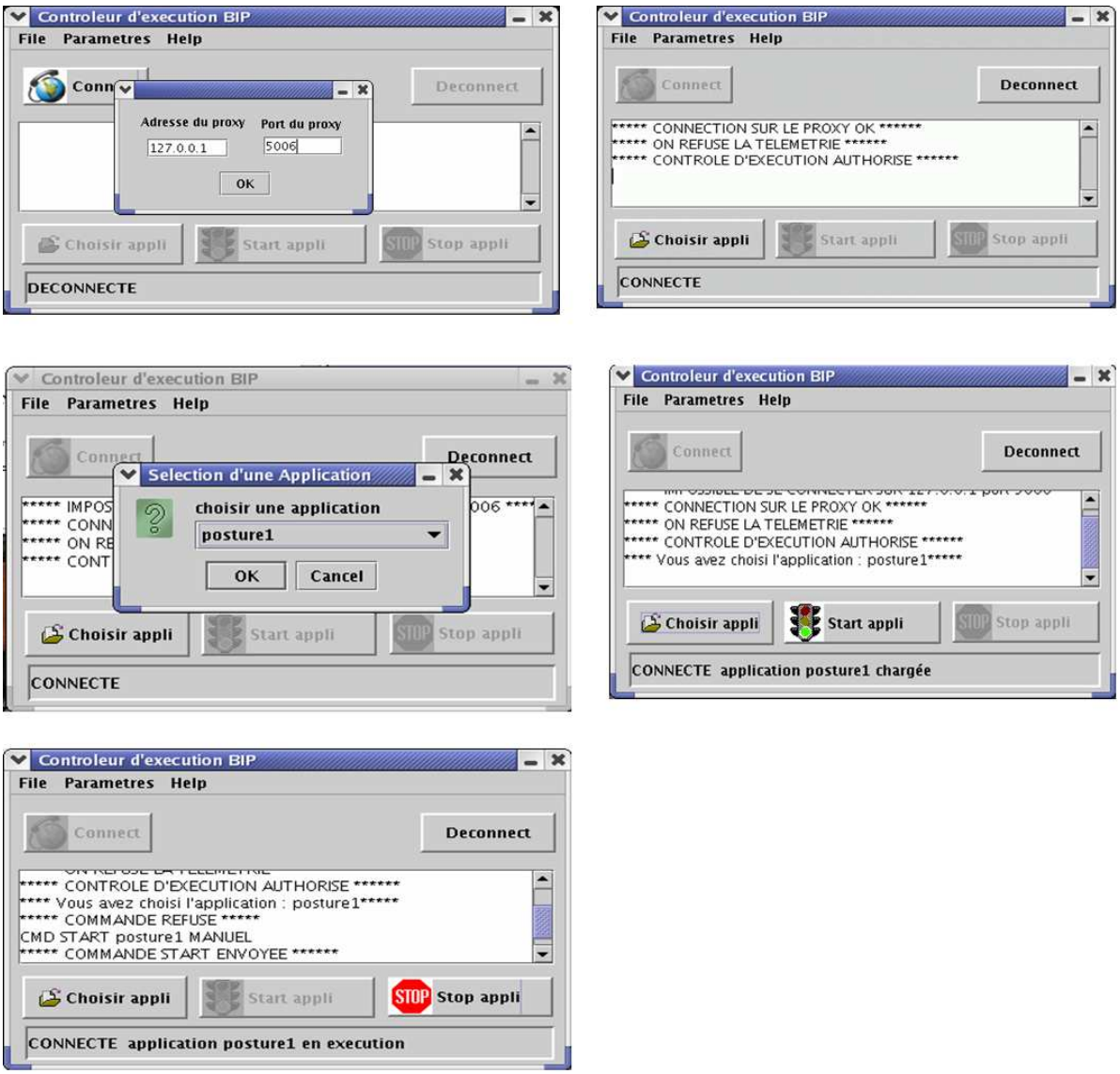

FIG. C.1 – Les différentes phases d'utilisation du client

Bibliographie

[1] COMAU . Pdl2 programming language manual. Rapport technique, ??, 2000.

[2] C. Azevedo and R. Pissard-Gibollet. Le contrôleur du robot BIP2000. Rapport de recherche, INRIA, 2000.

[3] G. Baille, P. Di Giacomo, H. Mathieu, and R. Pissard-Gibollet. L'armoire de commande du robot bipède BIP2000. Rapport technique, INRIA, 2000.

[4] B. Espiau and the Bip team. *BIP : A Joint Project for the Development of an Anthropomorphic Biped Robot.* Proc. Int. Conf. on Advanced Robotics, July 1997.

[5] D. Flanagan. *JAVA in a nutshell.* O'REILLY, 2000.

[6] M Guilbert. Intégration logicielle en vue d'une expérimentation en réalité virtuelle autour de la marche humaine. Rapport de stage, Ecole Nationale Supérieure des Techniques de l'Industrie et des Mines, 2002.

[7] S Jarde. Mise en oeuvre de commandes avancée sur le bip. Rapport de stage, ENSPS, 2002.

[8] M . loy, R. Eckstein, D. Wood, J. Elliott, and B. Cole. *JAVA Swing.* O'REILLY, 2002.

[9] A Luotonen. *Web proxy serveur.* Prentice Hall, 1991.

[10] F. Lydoire. Le simulateur du robot BIP2000. Rapport technique, INRIA, 2001.

[11] B. McLaughlin. *JAVA and XML.* O'REILLY, 2000.

[12] R. Pissard-Gibollet and K. Kapellos. **O**pen **R**obot **C**ontoller **C**omputer **A**ided **D**esign. orccad version 3.0β. User manual, INRIA, December 1998.

[13] J.M. Rifflet. *La communication sous Unix, Applications Réparties.* Ediscience International, Paris, 1996.

[14] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control : The Task Function Approach.* Oxford Science Publications, 1991.

[15] P. Sardain. Paramètres de BIP2000. Rapport technique, Laboratoire de Mécanique des Solides (LMS) de Poitier, 2000.

[16] The Orccad Team. *An Integrated and Modular Approach for the Specification, the Validation and the Implementation of Complex Robotics Missions*, volume 17-(4). International Journal of Robotics Research, April 1998. Special Issue on Integreated Architectures for Robot Control and Programming.

[17] P.B. Wieber. *Modélisation et commande d'un robot marcheur anthropomorphe.* Thèse de doctorat, Ecole des Mines de Paris, 2000.

[18] WindRiver Systems. *VxWorks 5.3.1 Programmer's Guide*, 1 edition, Mar 1997.

[19] WindRiver Systems. *VxWorks 5.3.1 Reference Manual*, 1 edition, Feb 1997.

[20] http ://www.inrialpes.fr/iramr/bip/ : le site public sur la réalisation du bipède.

[21] http ://www.esa.int/export/esaCP/index.html.

[22] http ://www.trasys.be.

[23] VIABLE Executive Summary, VIA-TRA-EXSUM.

[24] http ://www.comau.com.

[25] http ://www-lms.univ-poitiers.fr/robot/ : le site public du LMS de Poitiers.

[26] http ://www.inrialpes.fr/iramr/ : le site du service robotique de l'INRIA Rhônes-Alpes.

[27] http ://www.inrialpes.fr/iramr/Orccad/ : le site concernat le logiciel ORCCAD.

[28] http ://www.mcg.mot.com : le site fournissant des informations sur la carte MVME162-522A de motorola.

[29] http ://www.wrs.com/products/html/vxworks.html : Real Time Operating System de WindRiver Systems.

[30] http ://java.sun.com/ : le site incontournable de Sun (javadoc tutoriaux ...).

[31] http ://www.javafr.com/index.aspx : un site offrant beaucoup de codes sources java.

[32] http ://www.developpez.com/java/ : le site des developpeurs JAVA francais [codes sources, forum, examples ...].