Développement du logiciel de pilotage du SEMPO : cahier des charges, conception d'une maquette, modélisation UML des flux d'information et développement en langage JAVA de l'application finale

C.Prévost, O.Croce, O.Bernard et A.Sciandra

La première étape du développement a consisté en l'établissement d'un cahier des charges du logiciel de pilotage et la réalisation d'une maquette. La structure du système a été définie pour atteindre les objectifs en termes de fiabilité, d'évolutivité et de maintenabilité. Une maquette fonctionnelle en Tcl/Tk a été développée en respectant l'architecture identifiée dans le cahier des charges. Elle dispose des principales fonctions souhaitées pour piloter le bioréacteur et servira de support pour la réalisation du logiciel final. Les flux de données et l'architecture du système on été élaboré à l'aide d'une analyse UML. Le logiciel est en cours de développement en Java.

Ce travail a été initié par le stage D'O.Croce, et est maintenant réalisé par C.Prévost, ingénieur INRIA.

1. Présentation du système existant

Le dispositif expérimental de culture de phytoplancton fut développé voici plus de 10 ans ; les automates peu à peu installés sont assistés par des ordinateurs aujourd'hui dépassés. En effet il ne répondent plus à certaines exigences de mises en réseau ou de multitâches. Il devenait alors nécessaire de renouveler le parc informatique. Par voie de conséquence, les cartes électroniques de commande et d'acquisition permettant aux ordinateurs de piloter les appareils de mesures ont dû être remplacées également. Dans le même ordre d'idées, le logiciel de contrôle de l'automate analyseur de nitrates qui fonctionnait sur l'ancien système informatique ne fonctionne plus dans ce nouvel environnement.

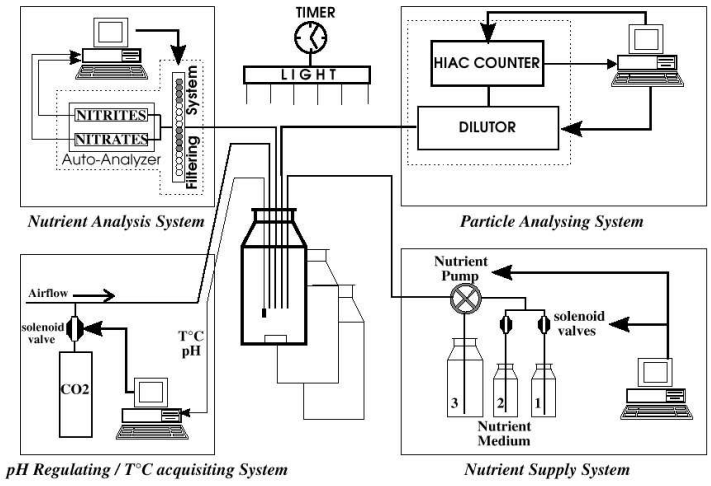

Fig.1. Schéma synoptique de l'actuel dispositif de culture phytoplanctonique. Les différents automates fonctionnent de manière indépendante. L'objectif du projet est de développer un logiciel *serveur* pour orchestrer ces automates.

Automate analyseur de nitrates :

Il fut pour cela nécessaire de transposer le programme Pascal initial en langage Delphi. Delphi représentant une évolution récente de Pascal, il semblait plus aisé et plus rapide de faire cette 'adaptation' du programme original (3200 lignes) que de le recoder totalement dans un autre langage.

Bien que Delphi soit proche du Pascal, de nombreuses bibliothèques ont été remplacées ou tout simplement supprimées. On pouvait quelquefois réécrire l'algorithme que fournissait la bibliothèque ou mieux trouver une bibliothèque de remplacement sur Internet. Il fallut recourir parfois à des astuces informatiques pour résoudre certaines erreurs ou malfonctions.

L'utilisation de nouvelles cartes de commande PCI a également nécessité de modifier profondément le script initial. En effet, les cartes précédentes de 32 bits traitaient directement l'envoi d'une valeur décimale. Les nouvelles cartes sont divisées en 4 octets. Il est donc nécessaire de convertir la valeur envoyée en valeurs binaires réparties sur les 32 bits, puis de les coder sur 4 octets.

Finalement, l'adaptation du programme de l'automate analyseur de nitrates a pris plus de temps que prévu, n'étant pas en mesure d'estimer *a priori* le nombre de problèmes 'astucieux' à résoudre.

La nouvelle version du programme s'exécute désormais sous Windows® sur console Dos (cf. Annexe 1.1.1.a).

Automate analyseur de particules :

Le programme de l'automate analyseur de particules s'exécute sous Dos également. On retrouve les mêmes limites d'utilisation que pour le programme précédent. Ecrit en C, il n'est pas envisagé de l'adapter à un autre environnement mais plus précisément de modifier certaines routines pour lui faire exécuter des taches supplémentaires.

Par exemple, pour permettre une communication entre le *serveur* et l'automate par l'intermédiaire de fichiers, le programme de l'automate doit gérer la création et la lecture de fichiers spécifiques.

Egalement, les données acquises par l'automate doivent être converties au format Ascii pour être utilisables. Ceci doit être automatisé pour autoriser une visualisation en temps réel.

Ces modifications n'ont pas encore été apportées au programme. Seule une rapide étude de la structure fut réalisée afin d'estimer la faisabilité de ce projet.

Réseau local :

Le *serveur* et les différents automates composant le système SEMPO vont être reliés par un mini réseau local. Un switch Nbase fera le lien entre les équipements et le réseau Intranet du laboratoire. Les résultats des tests du switch dépendent d'ailleurs de l'environnement de travail (Windows 95®, Windows 98®, Windows 2000®). Le premier ne permet pas le dépôt d'un fichier sur son disque dur s'il est réalisé par un autre ordinateur. Et il n'y a apparemment aucune gestion possible des droits d'écriture ou de lecture. Concernant les environnements plus récents, la gestion des droits est toujours critique (pour la version 98) mais ils autorisent tous les modes d'échange de fichier.

2. Analyse et définition des besoins

L'objectif est de développer de nouveaux outils logiciels pour le simulateur marin et d'améliorer ceux existants. Ces outils sont destinés à améliorer la gestion et la représentation des informations issues des automates dans le but de contrôler plus finement les différents paramètres du système.
Au départ du projet les automates fonctionnaient de façon indépendante, chacun collectant des données qu'il stocke sur disque dur. Une notion de synchronisation entre les machines est également à développer dans le but d'intégrer l'ensemble des flux dans un système de gestion et de contrôle global.
Deux parties étaient à développer : une base de données rassemblant les informations issues du système (automates) et un programme pilote utilisant cette base et des commandes utilisateurs. Tous les deux étant présents sur un serveur en réseaux avec les différents automates.

A partir du programme de pilotage, l'utilisateur aura la possibilité de :

- paramétrer les automates à distance,

- lancer des manipulations (démarrage de plusieurs automates simultanément)
- visualiser l'état général du système (affichage des problèmes, appareils en fonction, etc…)
- suivre en temps réel l'acquisition des données (graphiques)
- paramétrer les algorithmes de contrôle (seuils d'alertes, auto-calibration, etc…).

L'évolution du dispositif exige un renouvellement des moyens informatiques (installation de machines plus performantes). Ce changement doit permettre le fonctionnement des nouveaux programmes ainsi que la mise en place d'un réseau efficace.

- *Les modules du logiciel*

- le contrôleur : Il s'agit d'un module donnant des consignes aux automates actionneurs pour suivre les trajectoires imposées par l'utilisateur. Chacune des actions de ce module est répertoriée dans la base de données. Les informations nécessaires à ce module sont à la fois les paramètres de trajectoire données par l'utilisateur avec comme intermédiaire possible la base de données et les mesures fournies par les analyseurs (récupérées par interrogation de la base). Le contrôleur est sollicité par le signal.

- l'auto-calibreur : contrôle la manière dont les mesures sont effectuées (automates analyseurs). Il permet par exemple de régler le pas d'échantillonnage de façon optimale (augmente le nombres de mesures s'il y a de fortes fluctuations dans les valeurs précédentes). Les entrées dont il a besoin sont les paramètres de mesures donnés par l'utilisateur et les valeurs acquises par le système (via la base de données). Les opérations faites par l'auto-calibreur sont consignées dans la base. Ce module a besoin du signal pour être activé.

- détection des fautes et vérification du matériel : vérifie la cohérence des données et si les valeurs acquises ne dépassent pas les seuils spécifiés par l'utilisateur. Ce module effectue aussi un contrôle sur l'état du matériel, il vérifie par exemple que le niveau des cuves n'a pas atteint un niveau critique ou que le nombre de filtres restant est suffisant pour les prochains échantillonnages. Ce module utilise la base de données et nécessite un signal pour être activé. Ses actions sont stockées dans la base. En fonction du paramétrage de ce module il y aura activation ou non du module des alertes.

- les alertes : ce module regroupe les types d'alertes que le système devra mettre en place si nécessaire. Ces alertes auront pour but de prévenir le (les) utilisateur(s) en cas de problème lors d'une expérience. Il pourra s'agir par exemple d'envoyer des e-mails aux utilisateurs.

- paramètres de gestion des modules : Ce sont les paramètres de réglage des modules. Il s'agit en quelque sorte de paramètres avancés permettant aux utilisateurs expérimentés de modifier le comportement de chacun des modules.

- module de paramétrage: Regroupe l'ensemble des paramètres (configuration, trajectoire, gestion modules, mesures). Le paramétrage se réalise au moyen d'une interface graphique (IG). Ce module vérifie également la cohérence des paramètres pour chaque entrée et de façon globale. En début de manipulation, si tous les paramètres sont corrects, le module d'initialisation peut être activé.

- module d'initialisation : Il intervient pour débuter ou pour recommencer une manipulation. Il initialise à la fois le programme pilote mais aussi les automates, les synchronise et démarre leurs programmes spécifiques

- *Interface utilisateur-machine*

L'interface utilisateur-machine doit être simple et efficace. Le logiciel nécessite une prise en main rapide. Il est destiné à des utilisateurs n'ayant pas forcément de compétences en informatique. Il est donc impératif que l'interface soit intuitive et facile à utiliser. L'interface doit cependant proposer suffisamment de fonctionnalités. Pour cela, la solution est de présenter l'interface sous une forme simple avec des accès à des fonctionnalités plus complexes : paramètres par défaut pour un utilisateur non expérimenté et paramètres avancés pour un utilisateur confirmé.

L'interface graphique développée pour la maquette tente de satisfaire ces exigences. Le choix adoptée porte sur une interface graphique type Xwindows avec menus déroulant, boutons sélectionnables et

cadres d'affichage. L'interface graphique est composée de 5 écrans sélectionnables par des boutons placés en bas de l'écran, ainsi qu'une barre de menus déroulant.
Les menus déroulant auront les fonctions classiques que l'on retrouve dans la plupart des logiciels :ouvrir, fermer, quitter, enregistrer, aide etc…des options pourront être ajoutées par la suite en fonction des besoins.

- *Architecture de la base de données*

Les données contenues dans la base ne devront concerner qu'une seule manipulation lancée à partir du serveur. Il peut cependant y avoir les données de sous-manipulations chacune ayant lieu dans un chemostat différent. Cela permet au serveur de travailler sur une base peu importante. A la fin de chaque expérience la base sera exportée. On aura donc pour chaque manipulation un fichier contenant toutes les données, l'exploitation des données se fera à partir de ces fichiers. On peut différencier trois groupes de tables :

Les tables de mesures :

Ensemble des tables contenant les mesures effectuées par les automates. Une table correspond à un fichier de mesures, il existe donc autant de tables que de fichiers de mesures (importées). Leur structure peut varier, elle dépend du contenu du fichier importé

Les tables de paramètres

On trouve les tables de trajectoire et les tables de configuration pour les automates actionneurs, les tables de paramètres de mesures et les tables de paramètres de configuration pour les automates analyseurs. On a également les tables de paramètres de gestion des modules.

Les tables de journal

Table de paramètres de consignes, table des alertes, tables de commentaires. Tables contenant les informations sortantes du programme pilote : les consignes données aux automates par le module contrôleur, les alertes déclarées par le système, les commentaires entrées par l'utilisateur au cours d'une manipulation.

3. Conception et modélisation

Modélisation objet :

Un des objectifs de ce projet est de développer un ensemble modulaire et évolutif. Le simulateur est représenté par un ensemble d'unités, de fonctionnement proche (les automates) et requérant des méthodes similaires (gestion de fichiers, marche, arrêt...). L'analogie entre une représentation virtuelle du système et le 'monde objet' est directe. Souplesse et modularité sont de bons qualificateurs du développement orienté objet.

L'architecture générale du SEMPO est présentée par la figure 2 et l'annexe 1.2.a ; la figure 2 schématise les liens entre les différents modules et l'annexe 1.2.a les liens avec les appareils périphériques.

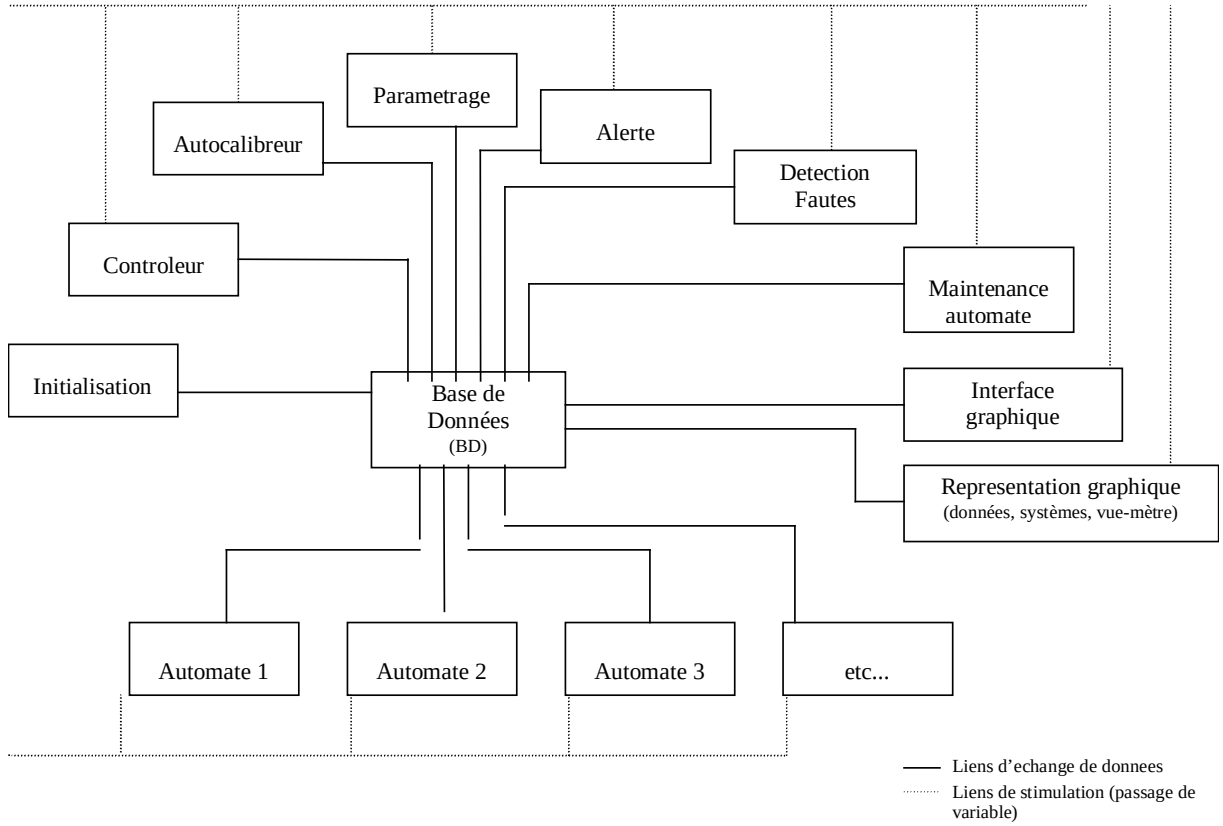

Fig. 2. Schéma général représentant les liens entre les différents modules SEMPO.

Le développement objet n'est pas toujours intuitif pour qui a, un temps soit peu, programmé en langage fonctionnel. Désirant se baser sur une véritable modélisation objet, il nous a semblé intéressant d'utiliser les standards UML.

Cas d'utilisation :

Ainsi après avoir détaillé avec précision les 'entrées' et 'sorties' de chaque module, le développement de 'cas d'utilisation', dans lesquels un module est placé dans différentes conditions de fonctionnement, paraissait le meilleur moyen de tester les caractéristiques du module et d'apporter de nouvelles fonctionnalités.

Les premiers cas d'utilisation développés s'insèrent dans le contexte d'une expérimentation en cours ; un certains nombre de modules interviennent de manière régulière (cf. annexe 1.2.2.a).

Si l'on isole l'activité de chaque module avec la base de données, cela autorise un développement plus complet des relations et processus de contrôle (exemple du détail du module Alerte – fig. 3).

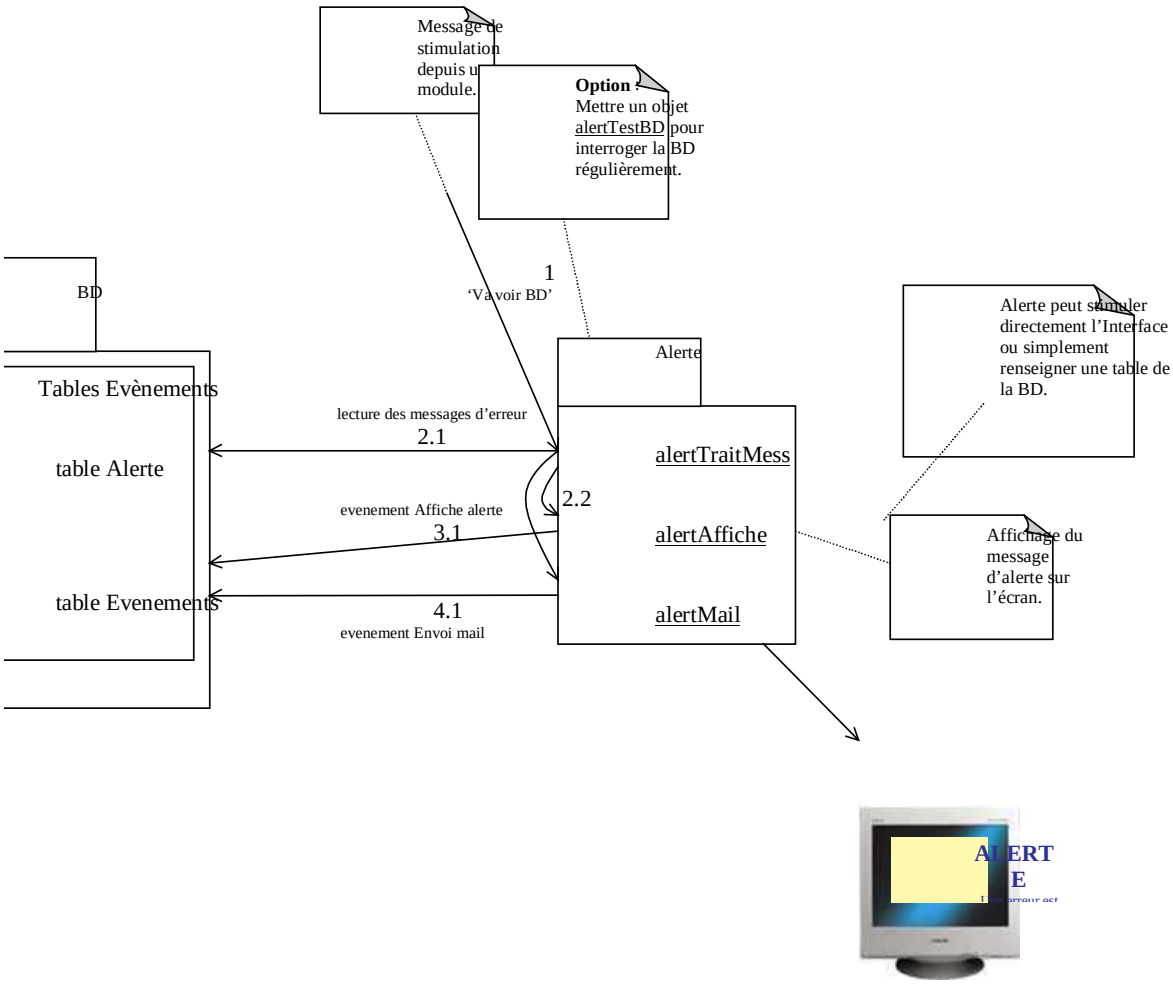

Fig. 3. Détail du module Alerte dans le cadre d'une expérimentation en cours.

Les opérations effectuées sont individualisées et numérotées par ordre chronologique. De nombreux commentaires peuvent également être ajoutés au schéma sous forme d'étiquettes ; nuances, explications, options ou questions restent ainsi liées au sujet auquel elles se rapportent. Cela favorise l'étude par étapes, en équipe de travail.

La base de données (BD) est le cœur du système. Elle sert de stockage de données résultats, commentaires mais aussi de relais d'informations destinées à d'autres modules. En privilégiant la relation unitaire BD/module, on simplifie le schéma relationnel pour les échanges de données et on favorise la modularité.

D'autres contextes que celui de l'expérimentation en cours ont été abordés ;

Le démarrage et le redémarrage (en cas de coupure de l'alimentation électrique par exemple) font intervenir d'autres modules ou les mêmes dans de nouvelles conditions (cf. annexe 1.2.2.b).

Avant cela (dans un fonctionnement classique), il faut configurer les automates, et ceci depuis l'automate lui-même ou depuis le serveur (cf. annexe 1.2.2.c).

La difficulté de cet ensemble informatique réside dans le fait que des données sont stockées sur le *serveur* dans la BD et que d'autres sont générées et stockées sur les différents automates. Il est essentiel de conserver un maximum d'homogénéité entre ces données pour assurer un fonctionnement cohérent. Les mises à jour constituent donc une activité à optimiser (cf. annexe 1.2.2.d).

Au cours de la réalisation de ces cas d'utilisation, la structure de la BD peu à peu s'établit, au travers des données à stocker.

Il convient de s'attarder sur cette phase de conception de façon à valider (théoriquement) les structures déployées. Cela rendra l'implémentation d'autant plus efficace, évitant des retours en arrière difficiles et une perte de temps conséquente.

4. Conception

La conception, qui consiste à apporter des solutions techniques, est intimement liée à l'analyse et fut entreprise en parallèle (modélisation UML).

Dans ce cadre comme d'un point de vue général, il sera également nécessaire d'adopter des standards pour le type de fichier et les formats de données utilisés pour la communication *serveur* / automates (cf. XML au paragraphe suivant). Le nom du fichier ou son extension renseigneront les objets chargés de la reconnaissance du type du contenu. Il pourra alors sélectionner l'opération de traitement adéquate.

Les messages d'information, envoyés par les modules vers la BD, seront codifiés pour permettre une gestion rigoureuse et allégée du suivi des états de chaque objet. Le but serait en effet, dans un premier temps d'enregistrer tous les comportements de chaque objets au sein de leur module respectif afin d'optimiser les opérations effectuées.

Cette phase de conception comprend également le développement des algorithmes au sein des modules de contrôle et de surveillance.

Le module *Autocalibreur* permet de réajuster les paramètres de mesures utilisés par l'automate de mesure selon une configuration prédéfinie. Plusieurs mesures étant effectuées, les données seront traitées par ce module grâce à un algorithme de contrôle. Cet algorithme, 'mou' ou 'réactif', devra déterminer la modification à apporter aux paramètres de mesures (exemple : diminution du pas d'échantillonnage).

Enfin, nous avons vu que le dispositif de culture expérimental actuel faisait intervenir divers logiciels de pilotage qu'il faudra gérer de façon homogène. Pour ce faire, différentes modifications sont à effectuer, et celles-ci doivent être décrites précisément au cours d'une phase de test et d'analyses des systèmes informatiques.

5. Réalisation et implémentation :

Implémentation :

L'étape suivante consiste en la réalisation de la programmation. L'application *serveur* est implémentée en Java pour ses potentialités et sa souplesse. Le 'temps réel' auquel nous aspirons est de l'ordre de plusieurs minutes ; la relative lenteur de Java n'est pas un inconvénient.

La base de données a été créée sur un serveur mySQL. Ce type de base de données est souple, rapide et permet un nombre d'enregistrements suffisants pour ce projet.

XML :

Automates et *serveur* SEMPO vont échanger un certain nombre de fichiers contenant des résultats de mesures, des données sur l'état des automates, des informations de suivi, des messages

d'alerte... Leur format étant arrêté, XML offre des outils indispensables à la mise en forme rigoureuse de données pour le transit d'une application à une autre.

Dans le cas de l'envoi d'un fichier incomplet, les 'balises de typage' d'informations encadrant les données empêcheront les confusions et permettront de valider ou non les informations (si incomplètes).

Tests et validation :

Ces tests permettent de réaliser des contrôles pour la qualité technique du système. Il s'agit de relever les éventuels défauts de conception et de programmation (revue de code, tests des câblages,...). On élaborera par exemple des tests d'échange de fichiers *serveur* / automates sur lesquels repose la centralisation des données de l'application SEMPO.

6. Conclusion

La conception du système est maintenant établie : la structure générale du système a été définie et les fonctionnalités requises pour le système déterminées. La maquette issue de la phase conceptuelle fonctionne et donne un aperçu de ce que pourrait être le logiciel final.

Les différents schémas permettent de prendre en compte l'ensemble des flux entrants et sortant des unités du système. Ils servent en outre à une bonne compréhension du fonctionnement du dispositif.

La structure du dispositif a été pensée pour satisfaire plusieurs exigences : robustesse, simplicité d'utilisation et évolutivité du système.

La notion de fiabilité est un élément important, le système doit être stable et devra être capable de réagir de manière cohérente à des situations atypiques. En cas de panne du serveur ou d'un des automates, le reste du système doit continuer à fonctionner isolément. L'acquisition des données n'est pas dépendante du serveur (programme de pilotage et base de données) mais reste sous le contrôle des automates.

De même certains choix ont été fait pour favoriser la robustesse, au détriment des performances. Par exemple, les données sont ajoutées à la fois dans la base de données mais aussi dans les fichiers des automates.

La simplicité a aussi été une priorité : le programme de pilotage doit avant tout rester un outil pratique. Grâce à des représentions simples (schémas du dispositif, graphiques), un utilisateur non informaticien doit facilement comprendre et utiliser le programme, il doit rapidement identifier les problèmes et réagir en conséquent.

La base de données MySQL a été implémentée et le codage JAVA est en cours; une première version (très rudimentaire) devrait être testée à la fin de l'année.

4. BIBLIOGRAPHIE :

Bernard, O., 2000. Développement d'algorithmes pour un simulateur d'environnement marin piloté par ordinateur (SEMPO). Appel à propositions. INRIA Sophia Antipolis.

Bernard O., Malara G., Sciandra A., 1996, The effects of a controlled fluctuating nutrient environment on continuous cultures of phytoplankton monitored by computers, Journal of Experimental Marine Biology and Ecology 197, p. 263-278.

O. Bernard, 1995, Etude expérimentale et théorique de la croissance de Dunaliella tertiolecta (Chlorophyceae) soumise à une limitation variable de nitrate. Utilisation de la dynamique transitoire pour la conception et la validation des modèles. Ph. D. Thesis, University Paris 6.

Croce, O., 2001. Logiciel de pilotage d'un simulateur expérimental de milieu marin : cahier des charges, conception et développement d'une maquette. Rapport DESS INRIA Comore.

HIAC Royco Division, 1993. Particle distribution analysis software. Operations manual, vers. 2.2. Pacific Scientific Company.

HIAC Royco Division, 1992. Syringe operated sampler, model 3000. Operations/maintenance manual, vers. 1.1. Pacific Scientific Company.

G. MALARA, A. SCIANDRA, 1991, A multiparameter phytoplanktonic culture system driven by microcomputer , Journal of Applied Phycology 3, , p. 235-241.

Malara, G. Automate de culture phytoplanctonique. Mode d'emploi. Document interne.

Muller, P.A., 1997. Modélisation objet avec UML. Ed. Eyrolles.

Ousterhout J., 1994, Tcl and the Tk toolkit, Ed. Addison Wesley Publishing Company.