Projet SEMPO – COMORE

Fin du contrat d'Ingénieur Associé INRIA

Logiciel de pilotage

d'un simulateur expérimental de milieu marin

Rapport d'activité

Ingénieur associé : Cédric PREVOST

Août 2003

SOMMAIRE

Introduction :

Ce rapport fait suite au premier rapport d'activité sur le logiciel de pilotage SEMPO, rédigé par Cédric Prévost (Projet COMORE) en avril 2002, détaillant les prémices du projet et son démarrage.

Le logiciel SEMPO est un logiciel de surveillance, de gestion et de contrôle adapté à un système expérimental de culture de phytoplancton en milieu ouvert. Lors d'une expérimentation, il assure le suivi de chaque automate du dispositif SEMPO, il rapatrie, dans une base de données, les données de mesures acquises par les automates, il alerte l'opérateur d'une anomalie de fonctionnement et peut agir directement sur l'automate pour modifier son activité.

Au terme des 2 années de travail qui m'étaient allouées pour la réalisation de ce logiciel au sein de l'équipe COMORE, je présente dans ce document la démarche suivie, les options prises et les difficultés rencontrées. La version actuelle, qui n'est pas la dernière je l'espère, montre quelques faiblesses que je signale en deuxième partie et auxquelles j'apporte quelques solutions. Cette version a malgré tout prouvé son utilité au cours d'une dernière expérimentation et a affirmé son potentiel d'adaptation à des systèmes informatiques variés.

1. DEROULEMENT DU PROJET :

1.1. Conception :

Il est nécessaire de distinguer la conception d'une part du logiciel, de celle d'autre part de la base de données. Ces deux parties sont intimement liées, point indispensable pour un fonctionnement correct du logiciel final, mais font appel à des notions radicalement différentes.

1.1.1. Le logiciel SEMPO :

Les bases de fonctionnement du logiciel SEMPO ont été définies grâce aux standards UML, permettant ainsi la réalisation de schémas illustrant les différents 'cas d'utilisations' du logiciel (cf. annexes 1). Ces schémas avaient l'avantage, outre celui de clarifier les idées, de permettre une présentation claire et relativement précise des futures fonctionnalités du logiciel aux autres membres du projet SEMPO. Les schémas de relatant pas les aspects techniques de la mise en œuvre, la compréhension du fonctionnement du logiciel ne nécessitait pas de connaissance spécifique en informatique. Il était alors de mon ressort de qualifier, en terme de faisabilité, les modifications que souhaitaient apporter les membres du projet.

Ainsi chaque module du logiciel a été décrit brièvement avec la dénomination des classes mises en jeu au sein d'un processus dynamique.

1.1.2. La base de données :

Dans les objectifs du projet, la base de données était destinée à l'enregistrement de données issues de mesures expérimentales mais aussi au stockage de valeurs de variables du système pour permettre un fonctionnement cohérent et convivial.

Cet objectif a été atteint suite à la conception d'une structure de base et à l'ajout de nouvelles tables et de nouveaux champs tout au long de l'implémentation du logiciel. La structure finale est illustrée par l'annexe 2 (détail de la structure de la BD).

1.2. Réalisation et implémentation :

1.2.1. Matériel et logiciels :

Le logiciel a été implémenté en langage Java 2 et testé sur la plate-forme J2SE (version1.3.1). Le logiciel de développement (IDE) utilisé est Borland JBuilder 4 Professionnel.

La base de données a été implémentée en MySQL (version 3.23.40) et tourne sur un serveur Apache (version 1.3.20).

Le package EasyPhp v.1.5 a été utilisé pour faciliter l'installation du serveur sous un environnement Windows.

1.2.2. Développement :

Les diagrammes UML présentés précédemment montrent les tâches principales que doivent exécuter les modules du logiciel. Chaque diagramme a donc été repris, détaillé à tous les niveaux (inter-classes, classes, attributs, méthodes) et codé en conservant à l'esprit l'aspect modulaire, évolutif et adaptable des classes créées.

Des tests réguliers ont été réalisés à chaque étape de la réalisation d'un module et les modifications éventuelles furent entreprises immédiatement après dans un souci d'homogénéité de l'ensemble du logiciel.

Des démonstrations et des discussions régulières avec les membres du projet me permirent, tout au long du développement, d'opter pour des solutions techniques d'un commun accord avec les futurs utilisateurs du système.

1.2.3. Communication logicielle :

Le système SEMPO étant destiné à une utilisation multipostes, à distance, il a fallu gérer différents problèmes de communication entre le serveur base de données, les postes clients et les automates.

La connexion entre le logiciel SEMPO et la base de données est assurée via le réseau local par un pilote JDBC pour MySQL (MM.MySQL 2.0.14). Le serveur est identifié par son numéro IP.

La communication entre le logiciel et les différents automates consiste, d'une part, en l'échange de fichiers de données ou de propriétés, et d'autre part, en l'envoi de codes du logiciel vers les automates. L'échange de fichiers est autorisé par le protocole Samba sous Windows. Il est dépendant du système d'exploitation et de la création de *'partages réseau*' ou de *'lecteurs réseau*'. L'envoi de codes est réalisé par 2 applications Delphi (créé pour résoudre la problématique) ; le premier est un module émetteur chargé d'envoyer sur le réseau local un code de 3 caractères à destination d'une adresse précise (numéro IP et port). Il est placé sur le même poste que le logiciel SEMPO et est démarré automatiquement. Le deuxième module, le récepteur, est chargé d'écouter sur un port défini et de démarrer le logiciel de l'automate ou tout autre logiciel répondant au code reçu. Il est placé sur l'ordinateur automate.

Au cours de la conception du système, XML avait tout d'abord été mis en avant pour standardiser l'échange de données entre les automates et le logiciel. Cette option a finalement été écartée car elle imposait une modification importante des logiciels automates.

1.2.4. Module AutoClic :

Pour s'adapter à chacun des automates, une nouvelle application Delphi fut développée quand la réalisation de certaines tâches restait impossible,. Nommée AutoClic, cette application est basée sur le principe de l'envoi de codes-touches (sendKey) pour simuler des pressions sur les touches du clavier pour réaliser une opération au sein d'un logiciel.

Cet outil est particulièrement intéressant dans le cas d'un automate dont le démarrage nécessite, outre le lancement du logiciel, la pression sur 1 ou 2 boutons et la saisie d'un nom de fichier par exemple. La séquence de démarrage est traduite sous forme de codes-touches et enregistrée dans la base de données. Au moment voulu, un fichier de configuration AutoClic contenant la séquence est créé par le logiciel SEMPO, puis ce fichier est lu et appliqué par le module Delphi pour exécuter le démarrage de l'automate. Ainsi, le démarrage et l'arrêt de certains automates peuvent être effectués à distance.

1.2.5. Profil idéal du logiciel automate :

Les automates dont le logiciel de contrôle a été réalisé par des membres du projet ont pu être légèrement modifiés pour résoudre certains problèmes d'échange de données ou de contrôle à distance. J'ai pu alors établir les principaux standards régissant le fonctionnement du logiciel automate 'idéal' et ses propriétés de création de fichiers :

Standards du logiciel automate :

- Fichier paramètres (définir lorsque celui-ci est lu, et il doit conservé la même longueur tout au long de l'expérimentation).
- Fichier d'arrêt de l'automate (à créer et à définir : 1 pour arrêt, 0 pour lancement).
- Format du fichier de données : un enregistrement par ligne.
- L'automate est un exécutable sous Windows.
- Les données enregistrées doivent être sous format texte (attention à certains types de caractères).
- Tout fichier ouvert doit être fermé pour être accessible par le serveur Sempo.
- Un fichier peut être incrémenté à chaque événement marquant de l'exécution de l'automate. Il pourra servir de fichier *Etat*.
- Un démarrage automatique du système est à prévoir.

Cependant, tous les logiciels automates et notamment les logiciels propriétaires ne suivent pas ces standards et il a été parfois indispensable de recourir au développement d'options très spécialisées pour s'adapter aux différences de fonctionnement.

Dans ce contexte, j'ai encadré un stage pour la réalisation d'une application destinée àla conversion des fichiers de mesures produits par l'automate HIAC (compteur et analyseur de cellules) en fichiers texte avec un pré-traitement des données.

1.3. Tests et validations :

Ces tests ont permis de réaliser des contrôles de la qualité technique du système. Il s'agissait de relever les éventuels défauts de conception et de programmation. Ces tests avec validations successives ont été conduits tout au long du cycle de développement et non àla fin pour éviter des reprises conséquentes du travail.

J'ai également pu profiter de la réalisation d'une véritable expérimentation utilisant le dispositif SEMPO, sur une durée de un mois et demi, pour faire fonctionner le logiciel dans des conditions réelles d'utilisation. Des erreurs informatiques, des erreurs d'adaptation, des erreurs d'enchaînement de tâches se sont déclarées et ont pu ainsi être corrigées. Egalement, de nouvelles situations expérimentales ont pu être appréhendées, donnant lieu à une extension du développement du logiciel pour permettre un fonctionnement correct dans tous les cas potentiels.

2. UTILISATION ET EVOLUTION DU LOGICIEL :

2.1. Utilisation :

2.1.1. Manuel utilisateur :

Le dispositif SEMPO est un prototype expérimental destiné à la recherche en biologie. Ses utilisateurs sont des scientifiques pour qui l'informatique doit rester un outil puissant mais simple d'utilisation. Mon investissement dans ce projet devait aboutir à la réalisation d'un logiciel à large potentiel d'adaptation, permettant néanmoins une prise en main simple et rapide. Comme dans toutes réalisations de ce type, j'ai passé beaucoup de temps sur l'aspect ergonomique, garant de convivialité, tout en agrémentant le logiciel de nombreuses options techniques, facilitant son emploi final.

Il était alors évident que l'utilisation du logiciel imposait la présence d'un *manuel de l'utilisateur*. Ce travail fait l'objet d'un stage que j'encadre actuellement. Ce manuel HTML hébergé sur le serveur sera disponible en ligne depuis le logiciel SEMPO et bénéficie déjà d'un moteur de recherche pour une utilisation plus efficace.

La création de ce guide m'a permis de mettre par écrit de nombreuses informations sur les détails de fonctionnement des modules du logiciel. Ils seront ainsi conservés et accessibles par tout utilisateur après mon départ.

2.1.2. Connexion à distance :

Le logiciel SEMPO ne dispose pas d'une double interface dédiée d'une part au serveur et d'autre part aux postes clients. Aux vues des conditions d'exploitation, du type d'utilisateur et du temps imparti pour le développement, j'ai décidé de me concentrer sur une interface unique utilisable sur tous les postes. Il en résulte un manque certain au niveau des sécurités notamment en ce qui concerne la base de données (cf. § 2.3. Développement à prévoir).

2.2. Maintenance :

Les matériels et logiciels utilisés actuellement pour l'exploitation du logiciel SEMPO sont amenés à évoluer. Il s'agit par exemple des systèmes d'exploitation, de la version de la machine virtuelle Java, de la version du serveur MySQL, des nouveaux logiciels automates prochainement déployés... Pour faciliter cette évolution, il me fallait développer un logiciel capable de fonctionner dans un environnement relativement standard ; ainsi au cours de la réalisation, le logiciel a été successivement installé sur des machines Windows (98, 2000, NT) puis sur des machines Linux. Outre l'aspect visuel légèrement différent, le fonctionnement reste identique. La connexion à distance s'accommode également des différents systèmes rencontrés (réseau local, réseau internet avec déclaration du poste client auprès du firewall).

2.3. Développement à prévoir :

Pour permettre des développements futurs du logiciel SEMPO en Java, le programme est accompagné des diagrammes type UML qui schématisent les tâches exécutées par un module donné, leur chronologie et les principales classes impliquées. Pour qui souhaite rentrer plus profondément dans le fonctionnement technique du logiciel, ces schémas donnent rapidement une vraie vue d'ensemble et donne les moyens de procéder à une petite modification ou plus largement de préparer l'intégration d'un nouveau module.

Ces illustrations sont de surcroît complétées par un document technique (Javadoc), consultable en ligne, listant les 50 classes utilisées, et fournissant pour chacune d'elles les propriétés de chacun des attributs (variables), constructeurs, méthodes. J'ai ajouté un commentaire explicatif, bien que succinct, pour chaque classe et chaque méthode.

Enfin, le code source du logiciel, largement commenté, finira de renseigner dans les moindres détails sur l'orchestration des opérations.

Le système SEMPO constitue un vaste dispositif faisant appel à de nombreux éléments caractéristiques. Le logiciel SEMPO est composé d'un grand nombre de modules pour prendre en compte tous ces éléments. Beaucoup d'options de fonctionnement, notamment des automatisations, ont été écartées pour cette première réalisation. Ce logiciel pourrait donner lieu à un travail de développement et d'amélioration pour quelques années encore. Dans cette optique, voici quelques thèmes, appréhendés, conçus ou dont le code a été réduit à son strict minimum, qui pourront faire l'objet de futures versions.

2.3.1. Visualisation graphique :

La version actuelle propose une visualisation, **par automate,** des données de mesures acquises au cours de l'expérimentation,. On sélectionne ensuite les champs de données que l'on veut voir figurer en abscisse et en ordonnée et on exécute la création du graphique.

Il serait plus pertinent de proposer une visualisation **par type de données** (ex : concentration NO3, PAR mesuré) procurant directement un graphe de suivi sans sélection répétitive des données d'abscisse et d'ordonnée. Naturellement, ceci implique une interface de configuration des graphiques, dans laquelle seraient prédéfinis les champs utilisés pour chaque type de données proposé.

Il s'agit donc d'une remise en forme de ce qui a été fait, sans modification des classes de dessin déjà créées (Graphes, Courbe, Axes...).

2.3.2. Tests de comportement :

Surveiller un automate à distance est délicat, voire difficile à réaliser si le logiciel de l'automate ne procure aucun signe d'alerte en cas de fonctionnement incorrect. C'est pourtant ce que tente de faire le logiciel SEMPO à moindre niveau. Il utilise un fichier déclaré 'Etat' comme témoin d'activité. Si l'automate ne modifie pas ce fichier régulièrement alors qu'il le devrait, cela laisse supposer une éventuelle panne. L'utilisateur peut alors en être alerté.

Il serait intéressant d'investir des efforts dans ce cadre pour améliorer cette reconnaissance d'erreur. Certes, les solutions sont souvent des astuces spécifiques aux caractéristiques de chaque automate mais cela fournit néanmoins une assistance non négligeable de l'expérimentateur.

2.3.3. Pilotage des automates actionneurs :

Initialement prévu, le pilotage des automates actionneurs a été seulement initié. Il repose sur la modification des paramètres du logiciel de l'automate. Cette modification est effectuée suite à l'analyse des mesures acquises et à leur comparaison avec des valeurs bornes. On distingue 2 actions : le contrôle qui est une modification de consigne et l'autocalibration qui est une modification des paramètres d'échantillonnage.

Un essai a été réalisé avec un automate 'pompe' en cours de développement.

2.3.4. Lancement général :

Dans la version actuelle du logiciel, chaque automate doit être démarrer individuellement. Il avait cependant été imaginé une option de démarrage simultané des automates voulus. Pour cela, une table de la base de données *Liste_autom_manip* a été créée pour accueillir la liste des automates à démarrer. Une seconde option était d'ajouter un champ '*manip*' avec la valeur 0 ou 1 dans la table *Liste_automates*, pour identifier les automates participant à l'expérimentation et ainsi ceux à démarrer automatiquement.

Il suffit ensuite de faire un lien vers les modules de démarrages séparés.

2.3.5. Redémarrage automatique :

Il est important de prévoir un système de redémarrage automatique du logiciel et des opérations de gestion en cas d'arrêt inopiné, suite par exemple à une coupure de courant. Un diagramme UML a été réalisé pour détailler la marche à suivre. Ce redémarrage est une option avancée du lancement général vu précédemment.

Un problème subsiste néanmoins :

☞ Comment passer la barrière de l'identification de l'utilisateur du système d'exploitation (Windows, Linux) ou comment lancer automatiquement un logiciel au redémarrage de l'ordinateur ?

2.3.6. Systèmes d'alerte :

L'objectif d'un système d'alerte est de prévenir l'utilisateur le plus rapidement possible. Un message à l'écran est quelquefois inefficace surtout s'il est noyé dans une longue liste d'erreurs et d'autres messages d'alerte. Pour cela, j'ai développé un moyen de catégoriser les erreurs et d'attribuer à ces catégories un degré de gravité. Les erreurs, lorsqu'elles sont émises, sont filtrées avant l'affichage en fonction du degré de gravité sélectionné par l'utilisateur. Il ainsi possible d'isoler les erreurs les plus importantes et d'y adjoindre des témoins lumineux pour rendre la gravité plus visuelle.

Cependant, l'envoi d'un e-mail ou même d'un SMS dans le cas d'un problème grave semblerait encore plus approprié et plus sûr pour l'utilisateur.

2.3.7. Un mini-réseau :

Au terme du projet, bien que le serveur et les automates seront placés dans la même pièce (le laboratoire d'expérimentation), ils utiliseront toujours le réseau Intranet pour communiquer. Une panne de ce réseau rend le système inopérant.

La solution réside dans le développement d'un mini-réseau, local au laboratoire, en utilisant un switch (NBase 100), et raccordé à l'Intranet. Le réseau du laboratoire serait ainsi isolé en cas de panne du réseau général mais toujours fonctionnel, et la communication au sein du système SEMPO serait alors conservée.

2.3.8. Sécurité du système et de la base de données :

Bien initialement prévu, la version actuelle du logiciel ne demande pas l'identification de l'utilisateur. Ceci permettrait cependant de limiter les options sensibles pour un certain nombre d'utilisateurs.

La base de données possède une table *Utilisateurs* prévue à cet usage. En initialisant quelques variables (noms, droits) au démarrage puis en affectant un test des droits utilisateurs à chaque option des menus déroulants, on obtient une première protection du système et des données.

La connexion du logiciel à la base de données est effectuée en 'anonymous'. Par la suite, il serait juste de réaliser une connexion avec un login et un mot de passe. En interdisant une connexion anonyme, on empêche toute intrusion dans la base de données.

La base de données n'est cependant pas trop vulnérable étant placée derrière le firewall au même titre que le réseau local.

2.3.9. Diagramme du dispositif expérimental :

Avant de lancer une expérimentation, il est nécessaire de configurer les automates participants puis le logiciel en fonction de la surveillance désirée pour le système. Un moyen rapide et simple pour l'utilisateur de vérifier ces paramètres de configuration repose sur la réalisation automatique d'un diagramme illustrant les automates, les chémostats et autres éléments mis en jeu. Des zones cliquables permettrait l'affichage de données de configuration suite à une requête dans la base de données.

En avançant plus loin dans le concept, on peut imaginer surveiller l'expérimentation en cours par un affichage dynamique des automates en fonction de leur état.

Cette réalisation pourrait être entreprise avec le logiciel AnyLogic, que COMORE vient d'acheter. Il permet de réaliser des applets grâce à un ensemble de bibliothèques et d'objets prédéfinis.

2.3.10. Une solution pour l'automate HIAC :

L'automate HIAC (compteur et analyseur de cellules) produit des fichiers binaires (avec le logiciel PDAS). Pour être exploitable, ces fichiers doivent tout d'abord être convertis en fichiers ASCII. Ceci doit être fait manuellement avec PDAS, lorsque l'automate n'est pas

en cours de mesures. Puis les fichiers ASCII doivent subir un nouveau traitement (effectué actuellement par une macro Excel) pour donner des valeurs représentatives et interprétables.

Afin de réaliser ces opérations de manière automatisée, j'ai prévu dans le logiciel, sur le panneau de contrôle des modules automates, une option qui permet de lancer l'exécution périodique de n'importe quelle application. Cette option démarrerait périodiquement une nouvelle application (à développer) qui se chargerait de convertir le fichier binaire avec PDAS et le module AutoClic, puis d'appliquer un traitement au fichier obtenu pour aboutir à un fichier interprétable. Ce fichier, dont le nom ou le répertoire est connu par le logiciel SEMPO, serait ensuite rapatrié dans la base de données.

Le diagramme (Fig.1) ci-dessous schématise le scénario d'automatisation des opérations de traitement des données.

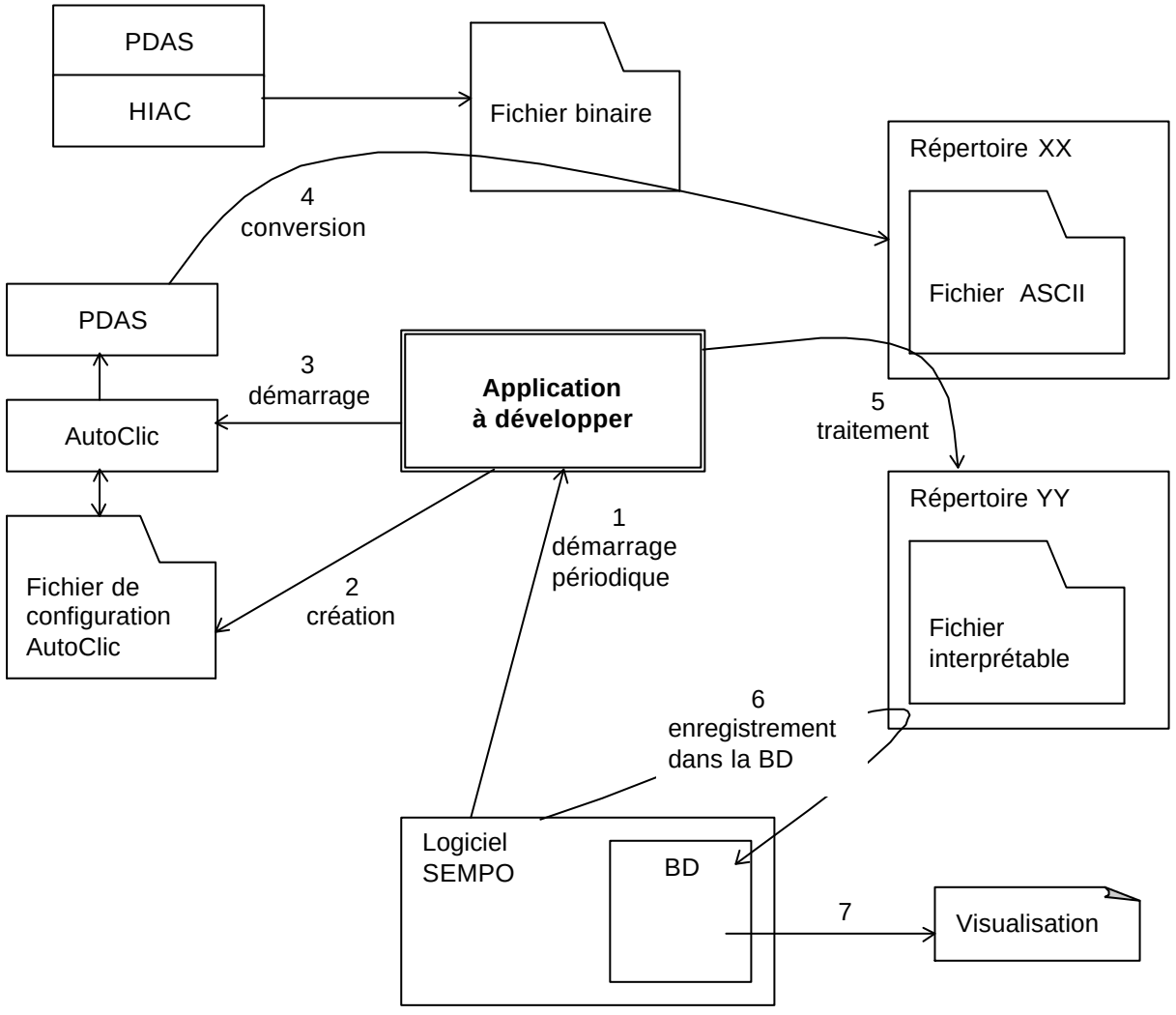


Fig.1. Diagramme illustrant l'automatisation de la conversion et du traitement des mesures HIAC et leur rapatriement dans la BD.

2.3.11. Support de commentaires utilisateurs :

Au cours de l'utilisation du logiciel SEMPO, un utilisateur peut souhaiter laisser un message à l'attention des autres utilisateurs (informer d'une modification, prévenir d'un comportement anormal...). Les outils le permettant ont été ébauchés et nécessitent un réel développement selon plusieurs thèmes :

- Cahier de laboratoire,
- Commentaires
- Maintenance
- Problèmes
- Dialogues

2.3.12. Optimisation de requêtes :

J'ai pu remarquer au cours des derniers tests réalisés sur le logiciel SEMPO que la tâche de mise àjour de la base de données était très longue às'exécuter (plusieurs minutes) lorsque la table de la base de données atteignait un grand nombre d'enregistrements (5000 lignes). Ceci provient de la comparaison effectuée entre les données du fichier de mesures à sauvegarder et les données déjà présentes dans la table de mesures. Il est cependant possible de désactiver cette comparaison puisqu'un processus d'identification des nouvelles lignes existe.

Malgré tout, on peut espérer améliorer les performances de la comparaison en optimisant la requête de la méthode *compareTable(String tableTemp, String tableDefinitive)* dans la classe *AutomRecupFichierComp*.

3. ACTIVITES COMPLEMENTAIRES :

3.1. Participation à un projet européen :

Olivier BERNARD, du projet COMORE, est le coordinateur scientifique du projet TELEMAC (TELEmonitoring and Advanced Control of high yield wastewater treatment plants). Il m'a ainsi permis de m'impliquer dans ce projet dès les premières rencontres des membres actifs et de l'organisations des tâches. Observateur plutôt que membre, j'ai pu alors appréhender avec recul les stratégies de déploiement d'un tel projet au sein d'un groupe cosmopolite composé d'industriels, de scientifiques (publics et privés), d'informaticiens et d'ingénieurs techniques (sociétés commerciales). Suivant le projet tout au long de mes 2 années de contrat, j'ai également été consulté, au titre de membre du projet COMORE participant, pour conseils et avis sur certaines réalisations informatiques (structure de base de données...).

3.2. Formations informatiques :

J'ai eu l'opportunité de suivre 4 formations officielles depuis septembre 2001, des formations qui rentrent de près ou de loin dans le cadre du projet SEMPO, nécessitant de multiples compétences :

Intitulé :	Date :	Lieu :	Durée :
Sécurité informatique	18/10/2001	INRIA Sophia	1 jour
Téléinformatique et réseaux	8/11/2001	INRIA Sophia	2 jours
Les fondamentaux de JAVA	8/04/2002	PCS Avolys	5 jours
XML	17/06/2002	PCS Avolys	3 jours

Outre ces formations professionnelles organisées de manière officielle, j'ai continué à me former sur de nouveaux sujets grâce aux compétences de professionnels de l'INRIA ou du CNRS.

Janet BERTOT (SEMIR, DREAM...), en tant que nouvelle 'tutrice', m'a initié à l'utilisation de CVS (système de gestion des versions de développement) au cours de plusieurs entrevues.

Je me suis également initié aux Systèmes d'Information Géographiques avec Caroline POCHO (CNRS-Villefranche-sur-Mer), bien que ceci soit plus loin des activités qui me concernent dans le projet SEMPO.

3.3. Encadrements de stages :

Le développement du projet SEMPO peut se décomposer en de nombreuses tâches de conception, de développement, d'inter-adaptation et de rédaction. Il est alors possible de déléguer certains travaux àdes étudiants.

J'ai ainsi eu l'opportunité d'encadrer dans le contexte de 2 stages :

- Le premier d'une durée de 2 mois avait pour objet la création d'une application pouvant convertir et analyser des fichiers de l'automate HIAC.
- Le second d'une durée d'un mois avait pour objectif la conception et la rédaction du manuel d'aide en ligne du logiciel SEMPO.

Ces deux stages m'ont permis de prendre plus de recul sur le projet et sur le travail déjà réalisé. Ils m'ont procuré de nouvelles contraintes d'organisation, testant ainsi mes méthodes de travail. J'ai ainsi pu appréhender la formation du côté du 'guide-formateur'.

3.4. Restructuration du site COMORE :

Le projet COMORE, comme la plupart des projets de l'INRIA, possède un site Internet présentant ses membres et décrivant ses activités. A la demande des membres du projet, j'ai procédé à la restructuration du site, lui donnant simplement un nouvel aspect, plus convivial.

4. CONCLUSION

Le projet SEMPO s'achève pour ma part puisque ma mission de réalisation du logiciel SEMPO arrive à son terme. Cependant, au cours des 2 années passées au sein de l'équipe COMORE, la nature innovante et pluridisciplinaire du projet m'a motivé chaque jour un peu plus. Le logiciel conçu et réalisé en équipe m'a apporté de nombreux réflexes de développement, une notable expérience en Java et en gestion de base de données. 2 années représentent également une période adéquate pour suivre un projet depuis son ébauche jusqu'à sa mise en œuvre. J'ai ainsi pu appréhender correctement la gestion d'un projet au sein d'une équipe avec ses contraintes de communication et de temps.

ANNEXES 1 : Shémas de type UML décrivant certaines opérations du logiciel SEMPO.

Expérimentation en cours :

Expérimentation en cours – Centralisation des mesures de l'Automate
Expérimentation en cours – Etat de l'Automate

Automate :

Configuration de l'automate depuis l'automate
Mise àjour des paramètres de l'automate depuis le serveur

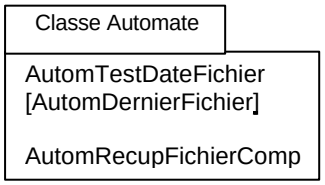
Erreurs :

Affichage et sauvegarde des erreurs survenues

Base de données :

Modules d'exportation des données
Modules graphiques ; Interface graphique, Visualisation graphique

Légende :

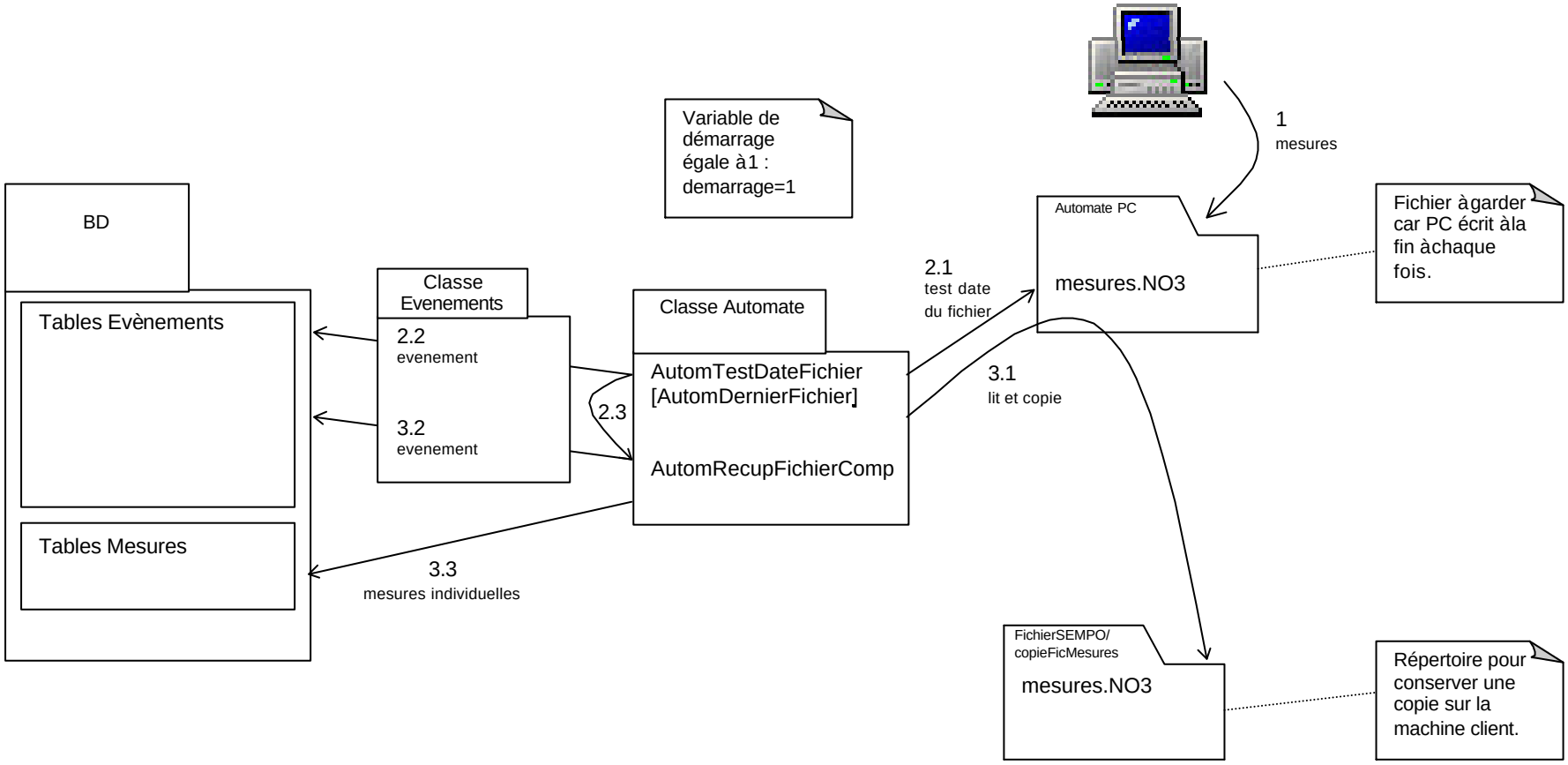


Classe initiale ou classe appelante :

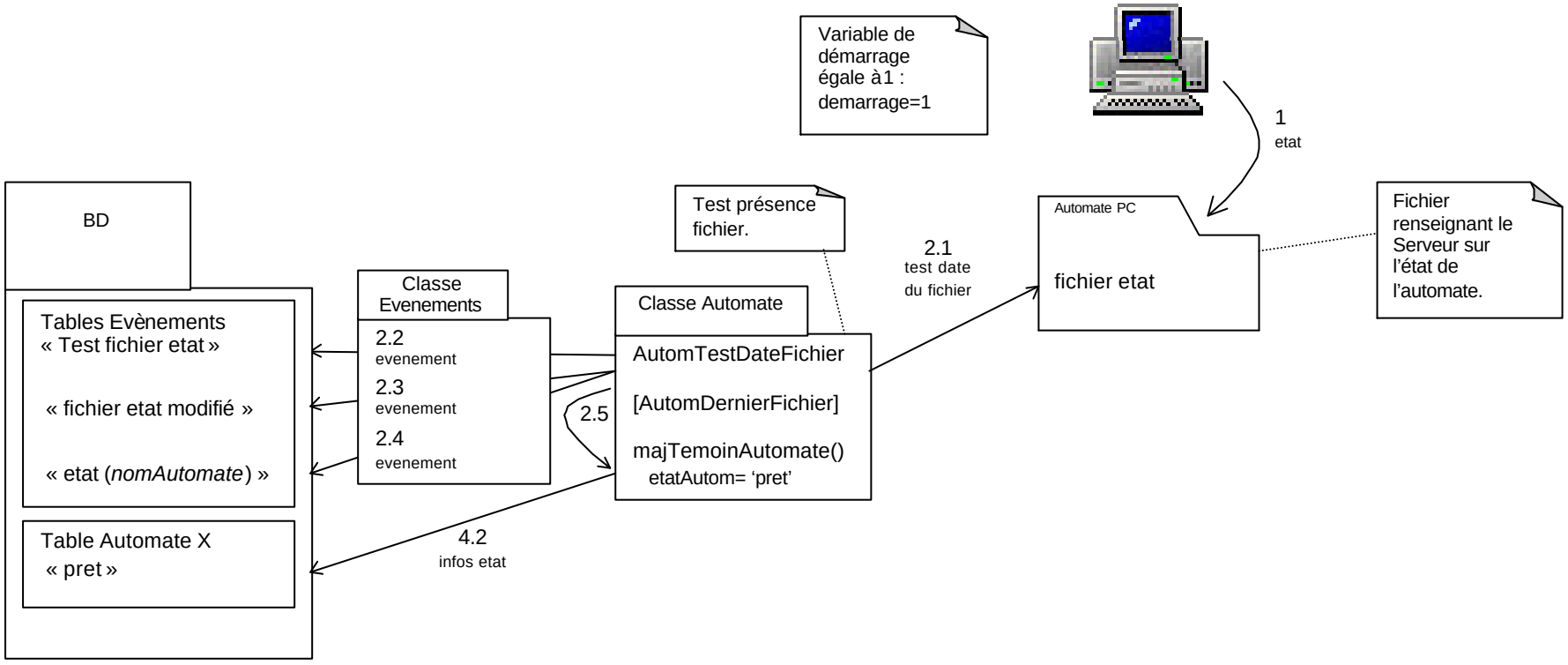
gère les opérations et appelle les classes participantes.

—— Texte et dessins de couleur grise : Tâche non réalisée.

UC : Expérimentation en cours – Centralisation des mesures de l'Automate (Procédure automatique)



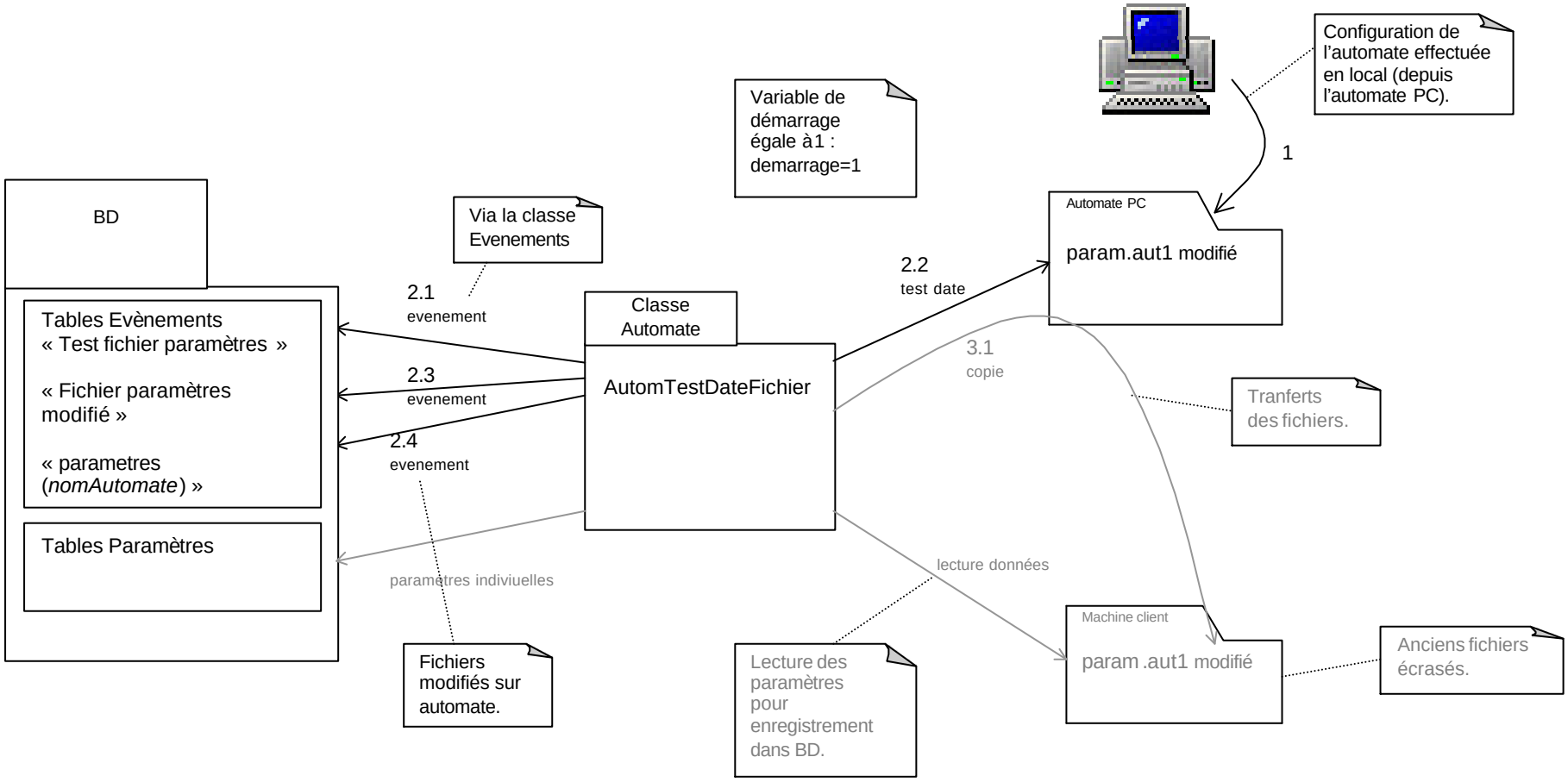
UC : Expérimentation en cours – Etat de l'Automate : automatique à intervalles réguliers.



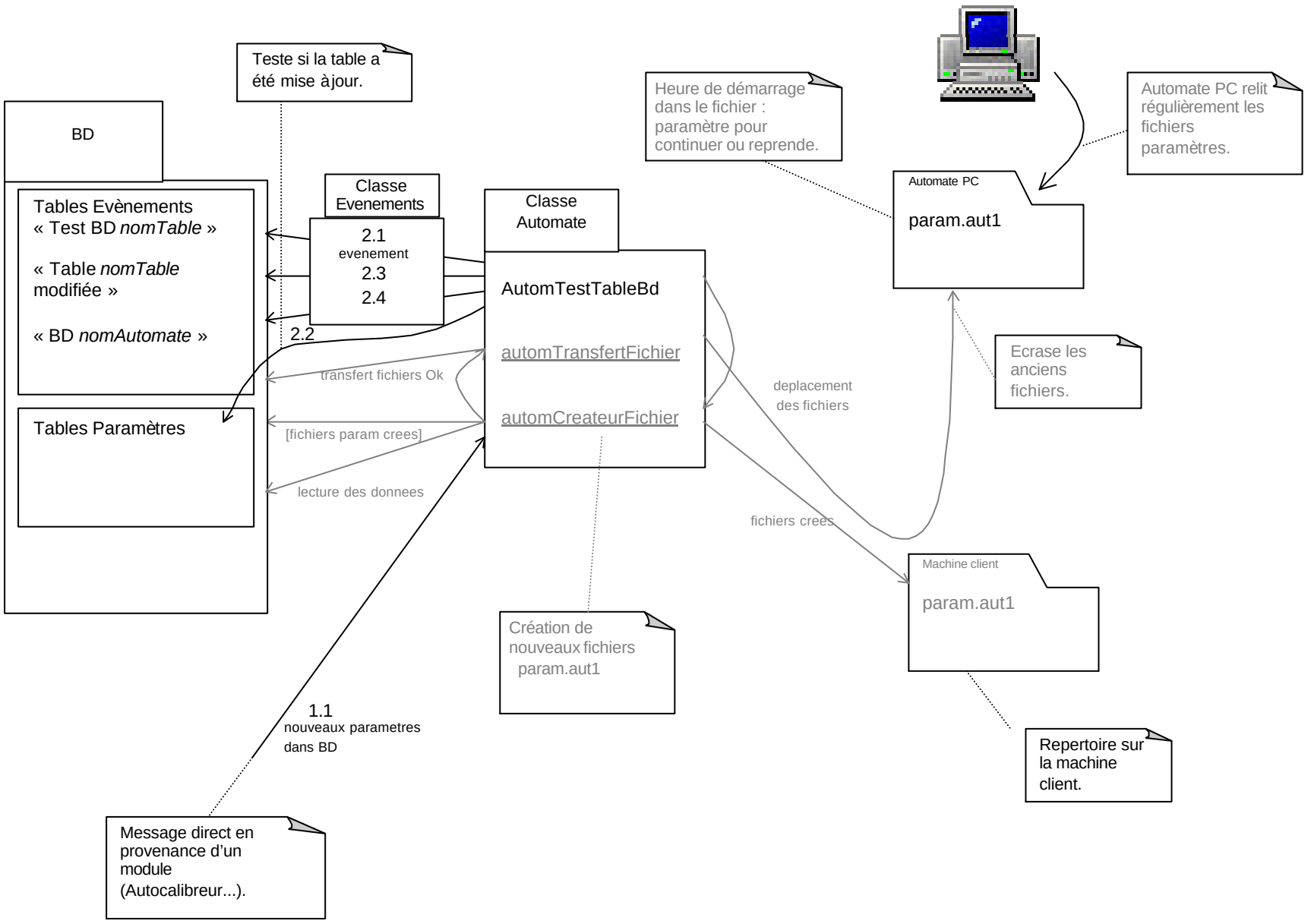
Notes :

- Lors du lancement du logiciel SEMPO, on peut imaginer une **initialisation automatique** de chaque module : **Test état automate** OK !

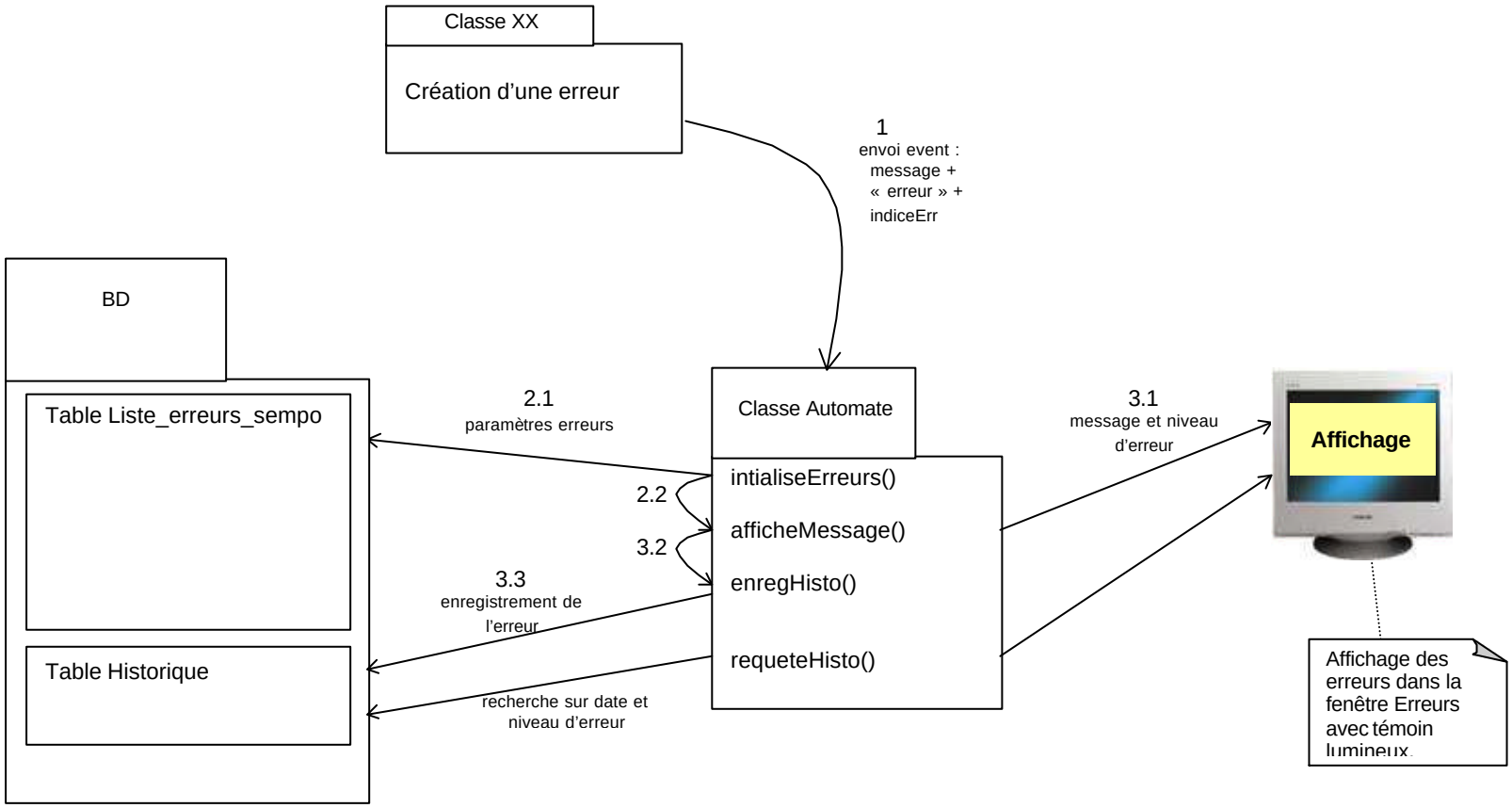
UC : Configuration de l'automate depuis l'automate.



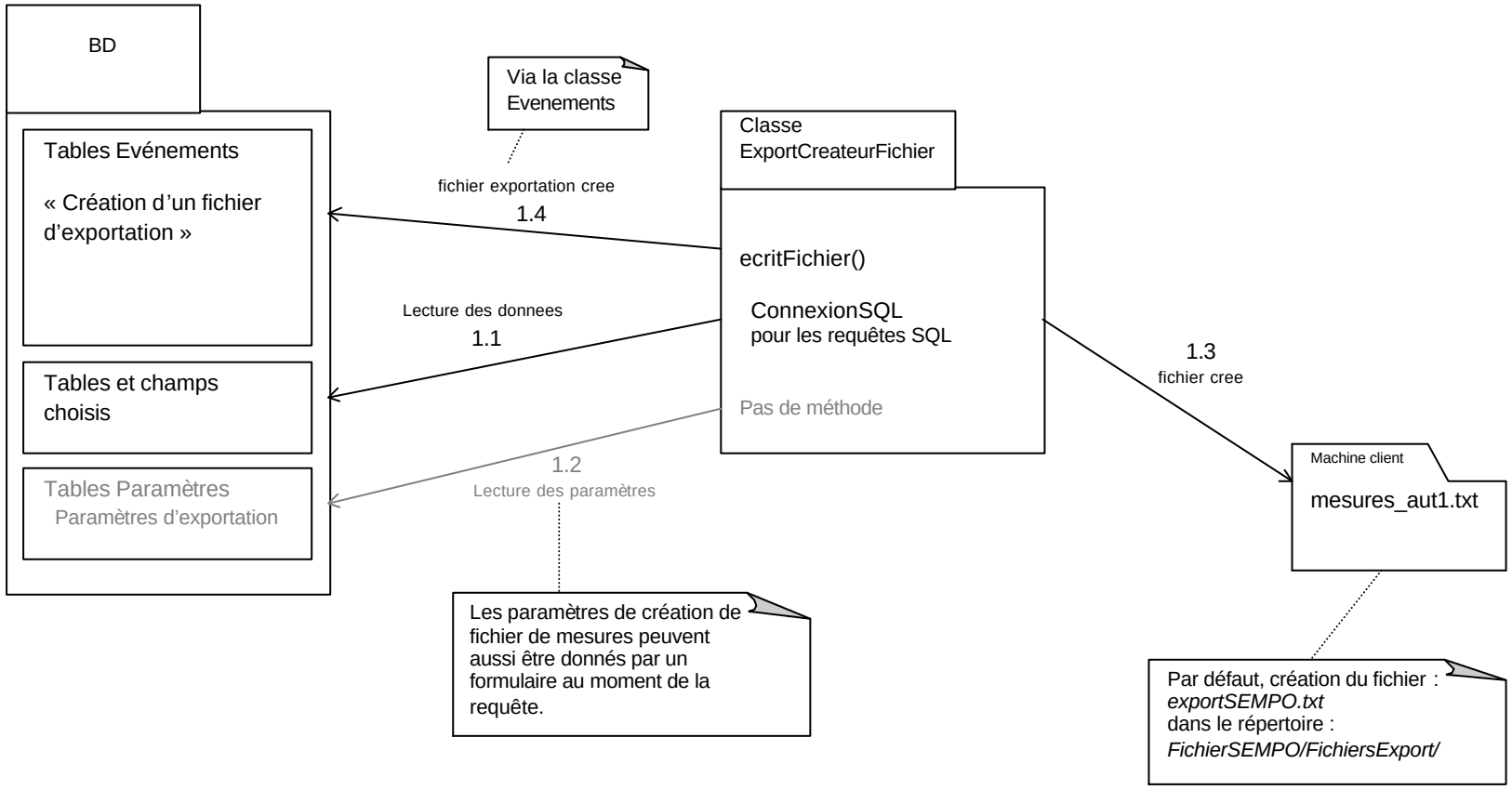
UC : Mise àjour des paramètres de l'automate depuis le serveur.



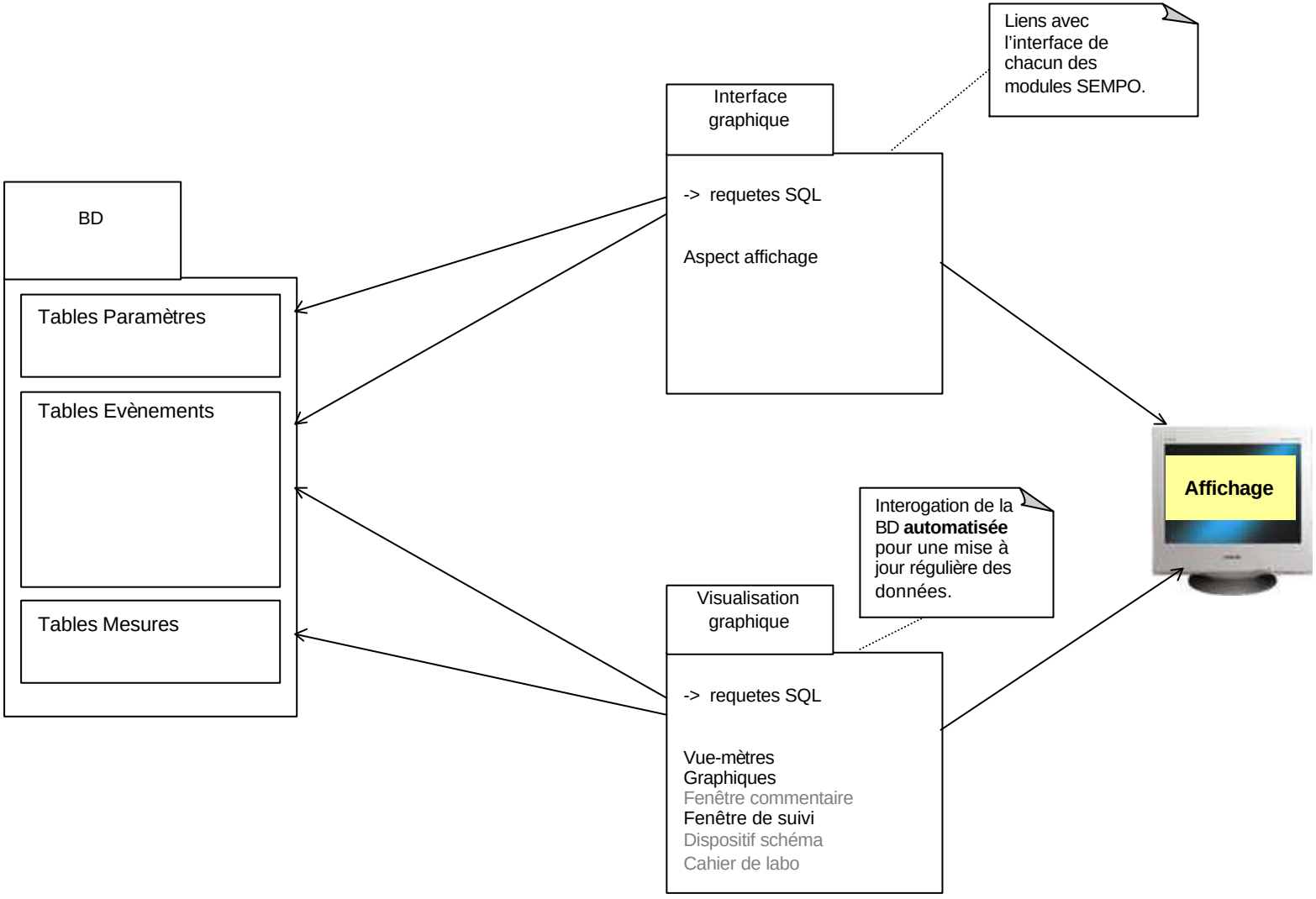
UC : Affichage et sauvegarde des erreurs survenues :



UC : Modules d'exportation des données.



UC : Modules graphiques ; Interface graphique, Visualisation graphique



ANNEXE 2 : DETAILS DE LA STRUCTURE DE LA BASE DE DONNEES SEMPO :

Tb Utilisateurs
-index_utilisateur
-utilisateur
-droits
-mot_passe
-observations
-maj

Tb Dialogues
-index_mode
-question_num
-commentaire
-maj
-observations

Tb Commentaires
-index_comment
-index_utilisateur
-date
-mode
-index_mode
-maj
-observations

Tb Cahier_labo
-index_cahier
-index_utilisateur
-date
-texte
-maj
-observations

Tb Problemes
-index_mode
-date_apparition
-date_disparition
-symptomes
-causes
-consequences
-solutions
-val_affectees
-commentaire
-maj
-observations

Tb Maintenance
-index_mode
-type
-commentaire
-maj
-observations

Légende

- Tables Administration
- Tables Documents
- Tables Evènement
- Tables Automates
- Tables Paramètres
- Tables Mesures
- Tables Modules cibles
- Tables Chémostats

Tb Evenements
-index_event
-provenance
-destination
-message
-maj
-observations

Tb Demarrage
-valeur_demar
-maj

Tb Historique
-index_histo
-message
-type
-indice_erreur
-date-erreur
-index_utilisateur
-autom_util
-observations

Tb Alerte
-index_alerte
-date
-provenance
-message
-type_alerte
-mode_alerte
-maj
-observations

Tb Liste_erreurs_sempo
-indice
-catégorie
-gravite_1a5
-maj

Tb Chemostats
-index_chemostat
-num_chem
-date
-....(caractéristiques)
-....
-index_lst_autom1
-index_lst_autom2
-...(ajout col. a chq nvl autom)

+ **Tables configuration** :
config graphiques
config exportations

Tb Liste_mes_autom
-index_lst_mes_autom
-index_lst_autom
-nom_tb_mes
-index_lst_param_autom
-maj
-sel_courante (0/1)
-observations
-champ_mes_compar1
-champ_mes_compar2
–plus_recent_fichier
-debut_nom_fichier

Tb Mesure_autom1
-index_mes_autom1
-nom_fichier
-delimiteur
-index_chemostat
-maj
-chp_valeur1
-chp_valeur2
-....
-observations

(*àcréer via l'interface*)

Tb Liste_automates
-index_lst_autom
-nom
-date_service
-description
-type
-icone_symbol
-adresse_IP
-nom_fichier_exe
-nom_fichier_etat
-plus_recent_etat
-debut_nom_etat
-nom_fichier_arret
-nom_fichier_interf
-maj
-observations
-freq_test_mesure
-freq_test_etat
-freq_test_param
-freq_maj_bd
-code
-port
-code_auto_clic
-demarr_auto_clic
-arret_auto_clic
-pause_avt_clic
-chemin_auto_clic
-box_demarr
-box_etat
-box_ficparam
-box_bdparam
-radio_ficmes
-radio_copimes
-radio_bdmes

Tb Liste_autom_manip
-index_lst_autom_manip
-index_lst_autom
-date_demar
-date_fin
-ordre_demar
-etat
-maj
-observations

Tb Automate1
-index_automate1
-index_lst_autom
-date_demar
-date_fin
-etat
-maj
-observations
-tem_cop_fic_mes
-tem_lance_autom
-tem_maj_bd_mes
-tem_fic_etat
-tem_fic_mes
-tem_fic_param
-tem_bd_param

(créée automatiquement)

Tb Liste_param_autom
-index_lst_param_autom
-index_lst_autom
-nom_tb_param
-index_param_automX
-nom_param (unique)
-maj
-observations
-nom_fichier
-separateur

Tb Param_autom1
-index_tb_param
-maj
-observations
-chp_valeur1
-chp_valeur2
-chp_valeur3
-chp_valeur4
-....

(*àcréer via l'interface*)