DOUBLE

PROGRAMME EUROPEEN DOCUP 2000-2006

PROGRAMME DE TRAVAUX D'APPUI AU DEVELOPPEMENT DE LA CANNE A SUCRE EN GUADELOUPE

CONTRAT D'ENTREPRISE DU CIRAD-CA
2e TRANCHE – Année 2003

RAPPORT D'ACTIVITES

ACCROISSEMENT DE LA PRODUCTIVITE DES CANNAIES
GESTION DURABLE ET VIABLE DES TERROIRS AGRICOLES DE GUADELOUPE

« ELABORATION D'OUTILS D'AIDE A LA DECISION POUR LA FILIERE CANNE-SUCRE-RHUM »

Chercheur : **Frédéricque BONNARD**
Pierre TODOROFF

JUIN 2004

Résumé

Jusqu'à présent, les activités en matière de développements informatiques en Guadeloupe étaient relativement restreintes. Or, actuellement le champ d'action de l'aide à la décision tend à s'étendre du fait de possibilités inédites : modèle de croissance, développement de la géomatique, informatique, connaissances du parcellaire et des pratiques accrues ... et de problématiques nouvelles : prévisions, rationnalisation et informatisation de la filière... Il est donc nécessaire de s'impliquer de manière beaucoup plus prononcée dans le développement et l'utilisation de l'aide à la décision, notemment en développant des méthodologies et des outils logiciels adaptés, répondant aux attentes, utilisant des données météorologiques, géographiques..., intégrant des principes agronomiques ainsi que des méthodes informatiques et de modèlisation.

Ce document fait le point sur les activités et orientations de l'aide à la décision.

Table des matières

1 Grandes lignes et orientations techniques 2

2 Avancement 4

3 Priorités et finalisations 9

1 Grandes lignes et orientations techniques

Voici les grandes idées directrices (ainsi que leur motivations) qui sont celles qui ont été adoptées pour l'aide à la décision.

1.1 Modularité

Quelques constations :

- un nombre croissant d'applications est développé au sein du programme canne : modèle de croissance (Simulex), modèle d'approvisionnement global (MAGI), logiciels d'irrigation, CASSIS, etc...
- les besoins auxquels ces applications répondent sont proches, du fait de domaines tous interdépendant, et d'un processus continu qui va du champs à l'usine (prévisions, croissance de la canne, irrigation, coupe, logistique)
- les solutions apportées sont parfois hétérogènes, les données aussi (base sous oracle, access, sig mapinfo, geoconcept, fichiers excel et autres formats ...)

Pour harmoniser tout cela, et créer une plateforme canne, il s'agit avant-tout d'effectuer un meilleur découpage, pour une plus grande modularité ; par exemple :

- une séparation des informations et traitements : utilisation de base de données pour stocker l'information, réutilisable pas différente applications (les bases de données fournissant également des processus de sauvegarde, réplication...)
- utilisation de préférence de protocoles, techniques, ou outils "passerelles" pour faciliter, la communication entre applications, et donc le couplage entre elles, tout en supprimant la dépendance envers certains formats...

1.2 Portabilité, interopérabilité

Autant que faire se peut, les outils préconisés seront ceux offrant le plus de souplesse en :

- ne se limitant pas un type de système d'exploitation (win,*nix...), de plateforme (x86...)
- proposant l'importation/exportation vers de multiples formats

Bref il s'agit d'éviter toute solution, nous conduisant dans une impasse en nous rendant dépendant de cette solution. Chaque outil utilisé, s'il devait être abandonné, ne doit bloquer qu'un minimum d'éléments.

1.3 Logiciels libres

Face à ces exigences, les logiciels libres semblent offrir un certains nombres d'avantages (dont ceux listés ci-dessus) du fait de leur définition (http://www.fsf.org/philosophy/free-sw.fr.html), à savoir :

- ils possèdent des fonctionnalités nombreuses et avancées car écrits par des passionnés, spécialistes ;
- un cout réduit voir dans la majorité des cas, nul ;
- efficacité : la possibilité de reprendre le logiciel si necessaire pour l'adapter à ses besoins (pas de perte de temps);
- une communauté autour qui fait que le support est large et rend le logiciel beaucoup plus fiable.
- la pérennité car le code source s'il est abandonné, peut être repris
- ils permettent la collaboration scientifique et le transfert de technologie (essentiels dans les contextes de travail du cirad) car ouverts (code accessible à tous) et portables.

  Pour résumer, les logiciels libres me semblent totalement en accord avec la mission de l'aide à la décisoon ainsi que celle du cirad.

2 Avancement

2.1 Rappel des principales actions à mener

Comme constaté lors des missions au sein du programme canne à Montpellier et à la Réunion, et suite à la première période passée en Guadeloupe, il s'avère avant tout nécessaire d'organiser tout ce qui concerne l'aide à la décision, de manière à éviter les redondances et faciliter la mise au point, le développement et l'évolution des données et applicatifs.
Les possibles actions de l'aide à la décision ont été formulées dans divers documents (cf RapportGuadeloupe 2002). Suite à la restitution début janvier 2003 (cf présentation), plusieurs axes de travail ont été jugés prioritaires pour l'année 2003 et trois actions ont été retenues ; dans l'ordre de priorité (du fait des interdépendances) :

- mise en place d'un système d'information (agro/carto/météo) : ceci correspond à la volonté de séparer l'information des traitements, et de les gerer Il s'agit ensuite de coupler des applications à cette base de connaissance. Parmi les applications phares Cassis et Mosicas, d'où les actions suivantes
- recodage de CASSIS
- refonte de Mosicas/Simulex

2.2 Le système d'information

Ce système d'information doit servir d'entrée à de multiples applications, aussi bien modèles divers (mosicas, modules de coupe.) qu'applications web, etc. Il comprendra, pour le moment, trois bases de données :

- une base de données météo
- une base de données agronomiques
- une base de données géographique

Plus généralement, toute donnée utile pour des applications en aval doit être disponible dans ce système d'information qui centralisera l'information. La création d'autre bases, d'ajout de variables sera alors aisée. Ces données seront essentiellement stockées sous la forme d'une base de donnée, cela pour faciliter leur stockage, gestion, accès, et s'affranchir de format trop divers, tout en restant portable.
Outre les données elles-memes, le système d'information pourra intégrer des routines de gestion des données : pré-traitement, calcul de variables intermédiaires ; mais aussi un système d'acquisition des données aussi automatisé que possible, notamment en ce qui concerne les données météo, puisque ce sont celles qui seront le plus souvent mises à jour ; la création de la base géographique ne sera faite qu'une seule fois et ne requiera, au pire, que quelques modifications de temps à autres ne necéssitant donc pas de réelle automatisation, si ce n'est la création de scripts permettant l'import à partir des différents formats d'origine, selon les couches d'information désirées. Quant aux données agronomiques,

leur acquisition se fera via une évolution de CASSIS.
Concernant la gestion des données, un certain nombre d'implémentations, d'outils, bibliothèques... ont été téstés afin de determiner ceux fournissant une réponse adéquate, chacun dans leur domaine. En voici la liste ;
Pour les systèmes de gestion de base de données :

- oracle (http://www.oracle.com)
- postgresql (http://www.postgresql.org)
- mysql (http://www.mysql.com)
- diverses autres implémentation

Ensuite pour pouvoir accéder aux données, plusieurs solutions ont été investiguées dans chaque domaine :

- tout d'abord des solutions de persistance serveur avec les serveurs d'applications et les EJB :
  1. jonas (http://jonas.objectweb.org) : serveur d'applications léger développé par le consortium objectweb
  2. jboss (http://www.jboss.org) : le serveur d'applications opensource le plus populaire
  3. andromda (http://www.andromda.org) : outil de génération de composants j2ee à partir d'un diagramme uml
  4. également, similaire en terme de génération automatique : xdoclet, middlegen
- face à certains problèmes de connectivité, une solution de persistence client a été préférée ; plusieurs ont été testées :
  1. db-ojb (http://db.apache.org/ojb)
  2. torque (http://db.apache.org/torque)
  3. hibernate (http://www.hibernate.org) : solution retenue

Les applications qui se connecteront à cette base et ne nécessitant pas une grande puissance de calcul (en particulier Cassis), seront en java, ce qui permet de pouvoir utiliser d'autres implémentations de bdd (grace à la majorité des outils/api utilisant jdbc ; autres intérêt : portabilité et toolkit graphique intégré...)
Par ailleurs, le design du framework base de données tel qu'il a été fait jusqu'à présent est le suivant : du fait des contraintes des clients (connexions modem, ponctuelles), l'idée d'avoir un serveur principal et plusieurs clients se connectant dessus pour effectuer des mises à jour (via par exemple des ejb, servlets...) a été abandonnée : les connections ne seront que ponctuelles pour "commiter" des modifications. La solution envisagée est une solution avec un serveur principal et des clients qui effectuent une synchronisation lorsque les personnes se connectent à internet. Pour accéder à cette base, une bibliothèque d'accès a été générée sur une base de persistance client à l'aide de Hibernate.
Toutefois, gérer plusieurs bases et leur synchronisation, pose le problème de réplication de bases. Ce problème n'a pas été résolu et peut l'être plus tard, car de toute facon pour l'application cliente (Cassis par exemple), ceci est transparent et donc la priorité a été donnée au développement d'une bibliothèque java pour accèder aux différents objets de la base agronomique. Cependant quelques solutions ont été envisagées :

- Utiliser une implémentation permettant nativement d'effectuer la réplication de bases ; mysql permet cela avec une notion de maitre-esclave, mais il reste à déterminer dans quel mesure la réplication est possible.
- L'utilisation d'une surcouche s'occupant d'effectuer la synchronisation : il existe une surcouche qui viens se greffer au dessus de jdbc, appelé c-jdbc qui permet de clusteriser différentes implentation de sgbd (supportant jdbc) entre eux ; c-jdbc est dans une phase de développement peu avancée et il est nécessaire de le tester plus avant si la solution native s'avère non utilisable

- en dernier recours une réplication manuelle : cela pourrait être un système de journalisation des modifications faites sur une base cliente, mais cela nécessite plus de travail, notamment au niveau de la gestion des mises à jour de la base serveur par les clients, mises à jour qui peuvent être incompatibles.

2.2.1 La partie météo

La base utilisée étant initialement postgresql, des procédures de calcul d'etp ont été portées de Oracle a postgresql. Cependant, au vu des problèmes posés par la synchronisation, les triggers auront éventuellement à être recodées en dernier lieu, puisque la base est sujette à évolution : si mysql permet d'effectuer la synchronisation, les triggers devront être adaptées.
Concernant l'acquisition, des éssais ont été fait seulement avec des stations Cimel, puisque qu'aucune station Campbell n'était encore accessible depuis une connexion téléphonique. En ce qui concerne les stations Cimel, il n'existe pas d'applications sous linux permettant d'en effectuer l'acquisition. L'utilisation de l'émulateur wine pour lancer le programme d'acquisition windows, n'a pas été totalement concluant... Une solution envisagée a été de reprendre le code qui avait été écrit à La Réunion, pour l'adapter, mais par manque de temps, l'utilisation de vmware (http://www.vmware.com) s'est avérée satisfaisante, puisque le support du modem dans vmware fonctionne et que l'acquisition peut ainsi etre effectuée.
La récupération des données de vmware vers la base de donnée peut se faire de plusieurs manières : stockage depuis cimsta dans une base access sous vmware/windows puis exportation vers postgresql via pgAdmin (http://pgadmin.postgresql.org). Sachant que cette solution n'est que temporaire, une autre solution est le passage des fichiers de données récupérés via samba (intégré dans vmware) et ensuite de parser celui-ci pour intégrer les données dans postgresql.
Ces procédures ont été testées avec des stations Cimel qui vont disparaitre a terme (remplacée par de stations Campbell, comme prévu par le projet météo Reinette), il faudra donc surement les adapter en déterminant la méthode et les outils/programmes utilisés pour interroger ces stations. La structure de la base est actuellement la même que celle utilisée à la Réunion.
Dans l'objectif de pouvoir mettre en place une procédure de récupération et la tester, une collaboration évidente avec le projet Reinette mis en place par P.Todoroff pourrait être établit (cf Projet météo 2003)
L'aide à la décision bénéficiera de tout l'ensemble des postes météorologiques pour effectuer l'acquisition des données (le projet météo prévoit de compléter le réseau actuel portant le nombre de stations à 11) et en échange des droits de consultation sur la base de données seront donnés.

2.2.2 La partie cartographique

Le choix de l'implémentation de la source de donnée cartographique n'est pas encore fixé. Il dépend en fait de tests de rapidité d'accès sur l'interface de Cassis, pour déterminer laquelle est la plus rapide. Ce choix depend également des toolkits utilisés ; pour une intégration dans l'interface java de Cassis, mon choix s'est porté sur le toolkit Geotools (http://www.geotools.org). Celui-ci fournit une api permettant l'accès à un grand nombre de sources de données, aussi bien fichiers que base de données : mysql (qui supporte nativement des objets géographiques) et postgresql (grace au module sig postgis, projet indépendant) entre autres. Il s'agit donc de déterminer si à l'utilisation, l'accès à une base de données géographiques (là aussi selon le sgbd utilisé) est beaucoup plus lent (à en être redibitoire) que l'accès aux données sous forme de fichier.
Des tests ont été effectués pour l'importation des données géographiques (des utilitaires fournis avec postgis par exemple postgis permette le passage de fichier vers base de données) et leur affichage grace à l'application mapviewer de geotools à été concluant concernant les fonctionnalités de geotools.

2.2.3 La partie agronomique

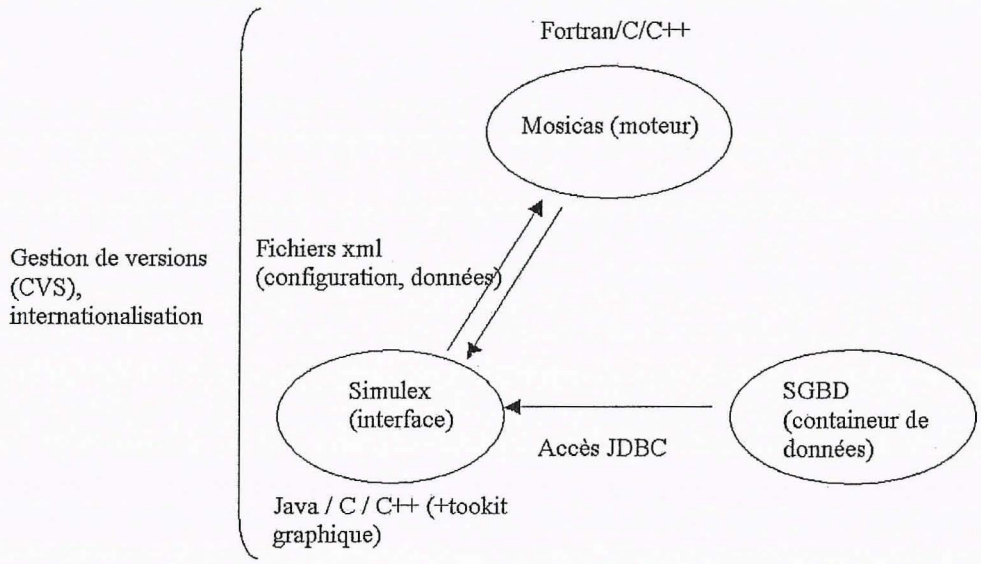
Le travail à porté essentiellement sur cette partie là. L'organisation de la base de données agronomiques a été reprise de la précédente version de Cassis et repensée. L'organisation des différentes entités a été formulée sous forme de diagramme de classe UML. La solution envisagée à l'origine pour gerer un serveur de données via des EJB a été abandonnée pour les raisons pré-citées, mais la solution consistant à faire appel à des techniques de persistance semble un moyen adapté de gerer une telle base. La solution de persistance serveur a été abandonnée au profil de la persistance client, et l'implémentation choisie est Hibernate.
La description UML de l'ensemble de la base de donnée agronomique a permis via hibernate de générer des classes d'accès pour l'accès persistant, fournissant un bibliothèque d'accès à la base.

2.3 CASSIS

Les travaux sur le GUI ont été entamé : cette ébauche de GUI a permit de voir que la bibliothèque d'accès à la base (pour les objets agronomiques) fonctionne (aussi bien via postgresql ou mysql).

2.4 La refonte de Mosicas

Ce travail était prévu en deuxième partie d'année. Suite aux échanges par mail concernant les aspects techniques informatiques concernant mosicas, l'aide à la décision à formulé quelques recommandations :



Quelques explications :

- Le sgbd est quelconque (bien qu'un véritable sgbd tel que Oracle/Postgresql/MySql est souhaitable comparé à Access plus orienté bureautique), JDBC (ou d'autres librairies) permettant un accès aussi bien à Access, Oracle, Postgresql, MySql JDBC est un driver d'accès sgbd en java (http://java.sun.com/products/jdbc/). Ainsi, que l'on demande à Simulex d'extraire des données

d'une base access ou oracle comme vous en avez, ou autres (les potentiels partenaires peuvent tourner avec mysql), ne modifie en rien le code. De plus JDBC rend des accès distants aux bases très aisés (et une solution de persistence, selon l'utilisation)

- Le passage de fichiers d'options, de configurations et de données est passé sous forme de fichier texte respectant un format xml (http://www.w3.org/XML/). Ce format très répandu est une méta-spécification, qui permet de définir simplement un format, une syntaxe à un fichier (de configuration par exemple). L'ajout future de nouvelles options de configuration ou données dans les fichiers textes, n'implique aucune modification du code concernant les options déjà existantes. D'autres part de nombreuses librairies existent pour parser des fichiers xml, ce qui permet, de récupérer les paramètres extrêmement simplement, de vérifier la syntaxe, etc... Les fichiers xml peuvent aussi être réutilisés par des applications webs.
- Pour l'interface graphique qu'est Simulex, le fait de la coder en Java (http://java.sun.com/) lui assure une très grande portabilité, aussi bien sur des OS de versions différentes (win 95, 98, 2000, XP), des OS différents (Windows, Unix, Linux...), qu'à des architectures différentes (pc, stations, hp, sun). Un développement en VB ou Delphi me semble inadéquat (pour les raisons qui ont amené notamment à l'abandonner pour Mosicas). La portabilité sera nulle, et ce ne sont pas vraiment des langages destinées à des travaux logiciels très importants (suivi, internationalisation, problème avec l'évolution des versions de ces environnements de développement). Le C ou C++ sont envisageables, mais nécessitent le choix d'un toolkit pour développer les interfaces et le travail de développement peut être plus important.
- Concernant Mosicas, les développements continueront en fortran même si cela ne me semble pas une solution viable à long terme (problème en termes de génie logiciel (maintenance, débuggage), de normes multiples, de compilateurs veillissants, portabilité). Il est donc important de commencer à réfléchir à un portage vers un langage moins problématique.
- Enfin, des aspects comme la gestion des versions avec des outils tels que CVS (http://www.cvshome.org/) et la prise en compte de l'internationalisation sont à intégrer dans le processus de développement dès le départ.

Ce design qui rejoint celui de JF Martinet en terme d'architecture,

- permet de séparer données, traitements, post-analyses apportant les avantages inhérents à cela : modularité et donc évolutivité
- n'impose pas de formatage spécifique de données, assurant l'interconnexion avec différents sgbd et la réutilisation des fichiers xml très simplement.

3 Priorités et finalisations

Sur les trois principaux axes, le travail sur mosicas n'ayant pas été entamé, et du fait de dépendances, il s'agit essentiellement de terminer la mise en place du système d'information et l'adaptation de Cassis ; à savoir dans l'ordre de priorité croissant :

3.1 La synchronisation des bases

D'après les trois solutions envisagées, trouver la plus adaptée, pour pouvoir synchroniser des bases clientes sur une base centrale.

3.2 Acquisition et traitement des données météo

Même si plusieurs voies ont été investiguées pour l'acquisition via une station Cimel, l'acquisition sur des stations Campbell reste à mettre en place, puisque c'est à terme celle qui seront installé et utilisé, et aucune station de ce type n'étant disponible (cependant l'INRA a localement installé une telle station et un stage s'est déroulé pour tester les interrogations...)
Les solutions envisageables :

- acquisition via vmware (fonctionne pour Cimel et facilement adaptable pour Campbell)
- acquisition native sous linux : reprise (à partir de ce qui a été La Réunion pour Cimel) ou écriture d'un logiciel d'acquisition (concernant Campbell, un tel logiciel a été développé en Corée pour une étude sur le riz... http://epilab.snu.ac.kr/rice/blast-detail.html)

La première solution est donc celle qui a été utilisée mais le but serait de tendre vers la deuxieme pour plus de sureté (ce qui necessite plus de temps).
Eventuellement, l'on devra adapter les triggers de calcul d'ETP à Mysql (si Mysql devait être choisi en raison du problème de synchronisation).

3.3 L'interface Cassis

La bibliothèque d'accès à des objets agronomiques stockés sur une base fonctionnant, il s'agit maintenant :

- tout d'abord, de coder l'équivalent de l'interface de la précedente version de cassis.
- rajouter des fonctionnalités cartographiques grace au toolkit geotools.