JMMC-MEM-2300-0003

Revision 1.0

Date: 06/03/2007

JMMC

Bilan sur les essais de couplages entre Yoga et une interface graphique Java

Authors:
Guillaume Mella <Guillaume.Mella@obs.ujf-grenoble.fr> — LAOG/JMMC

Author: Guillaume Mella Institute: LAOG/JMMC	Signature: Date: 01/03/2007
Approved by: Institute:	Signature: Date:
Released by: Institute:	Signature: Date:

Change record

Revision	Date	Authors	Sections/Pages affected
	Remarks		
0.1	01/03/2007	G.Mella	all
	Premier brouillon (envoye par mail le 6 mars)		
0.2	06/03/2007	G.Mella	all

Contents

List of Tables

List of Figures

1 Introduction

1.1 Object

Ce document rassemble le maximum d'informations et fait le bilan d'essais pour la realisation d'une interface graphique pilotant le logiciel Yoga. Ce travail conjoint a demarre suite a l'accord unanime lors de la reunion technique logiciel du 26 octobre 2006[2]. Une esquisse d'architecture et un mini-plan de travail ont ete definis lors de cette reunion.

Le bilan fera etat du suivi du plan de travail et d'une analyse technique sur les developpements realises au cours de ces 4 derniers mois vis-a-vis de l'architecture envisagee initialement.

Une page de travail wiki http://www-laog.obs.ujf-grenoble.fr/twiki/bin/view/Main/GMModelFittingDevelop Premiere_proposition_d_architect(acces restreint) a ete mise en place pour rassembler les elements importants pour le developpement entre le GL et GG.

1.2 Reference documents

[1] JMMC-MIN-2000-0001, Minutes de la reunion technique logiciel - Lyon, 17 octobre 2006

[2] JMMC-MIN-2000-0002, Minutes de la reunion technique logiciel - Grenoble, 26 octobre 2006

2 Plan de travail

2.1 Lancer un fit avec la premiere version du fichier de description

Valide depuis mi-novembre 2006. A ce moment la une solution combinant la generation automatique d'un fichier descripteur a partir d'une representation xml d'un modele permettait l'execution d'un batch yorick.

2.2 Fixer une version 1.0 du fichier de description

Le fichier *yoga/LITpro/model/model_ template.mdl* rev1.1 du 4 decembre 2006 commente et fixe une version fonctionnelle. Cette version sert de base aux definitions d'interfaces logicielles.

2.3 Fixer un mecanisme d'interruption stop+dump dans Yoga

Ce point n'a pas ete traite par le GL.

Je considere cela non critique s'il s'agit uniquement d'interrompre le traitement en cours (ce qui implique que l'on pert ce qui vient d'etre calcule). Dans ce cas le GUI peut a tout moment interrompre un traitement sans que cela necessite de nouvelle fonctionnalite du cote Yoga. Cela ne signifie pas qu'il ne faille pas implementer le dump dans le futur. Le degre zero n'a cependant pas ete implemente completement cote GUI de part son absence d'utilite au cours des essais. Cependant avec la solution propose le cas extreme ou le programme bouclerait a l'infini peut etre ecarte par la definition d'un timeout.

2.4 Obtenir la liste des modeles disponibles dans Yoga

Pour l'instant la liste des modeles n'est pas construite par Yoga de maniere dynamique. La difficulte majeure est d'avoir un mecanisme extensible qui prendrait en compte facilement les modeles utilisateurs. En attendant la mise en place d'un mecanisme dynamique, un fichier statique liste les modeles pris en compte par Yoga dans sa configuration.

3 Architecture generale

Nous avons imagine une solution qui fonctionne en mode client serveur. Un GUI Java peut etre telecharge sur le poste client. Il effectue de maniere autonome la plupart des actions hormi l'etape d'ajustement de

modele. La boucle principale consiste a envoyer sur le serveur un fichier xml contenant toutes les informations necessaires a l'ajustement de modele. Lorsque un fichier xml embarque des donnees binaires (oifits), celles ci sont encodees en base64. Le resultat xml permet de mettre a jour le GUI et d'informer du traitement effectue par le coeur du logiciel : yoga.

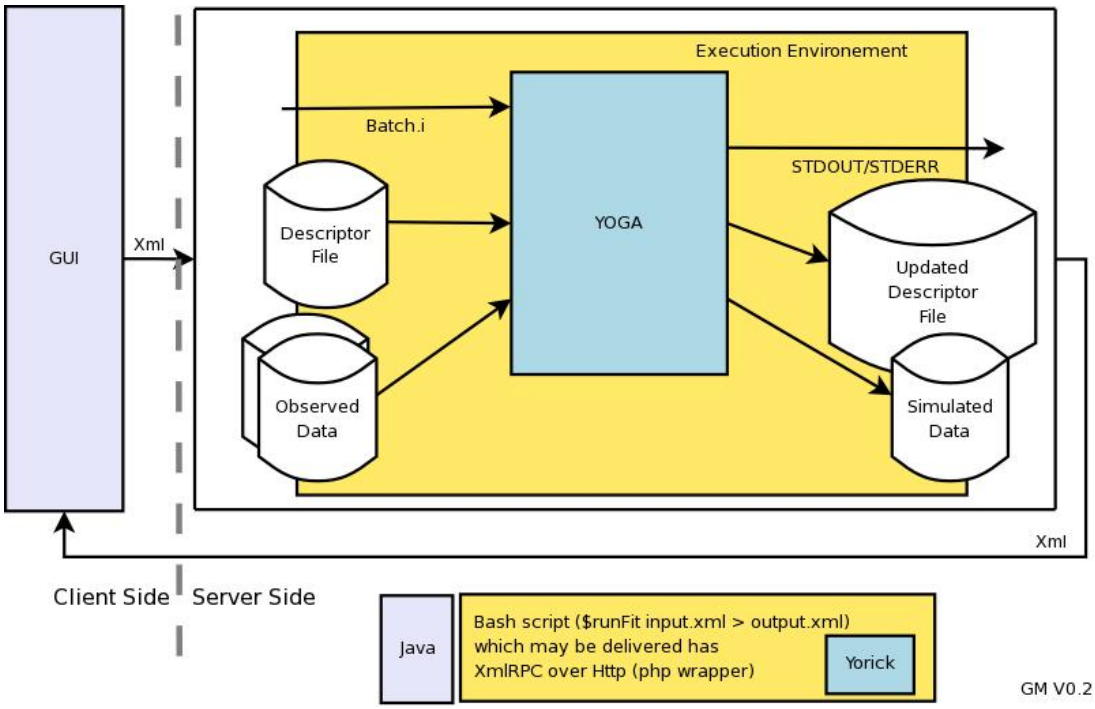

Figure 1: Architecture generale

4 Partie specifique Yoga

La seule modification apportee au programme pour les besoins de l'interfacage est la transformation de hash-table yorick sous format XML.

```
RCS file: /home/cvsroot/yoga/yoga/OIlib/yorick/oidata.i,v
...
----------------------------
revision 1.21
date: 2007/02/16 07:05:04; author: jmmc;  state: Exp;  lines: +8 -4
Add empty element in xml associeted to yorick void objects
...
revision 1.18
date: 2007/01/08 08:56:54;  author: jmmc;  state: Exp;  lines: +65 -1
Added one draft definition of oi_show_in_xml
...
----------------------------
```

Le passage par xml a permis l'exploitation immediate des variables yoga en dehors de l'interpreteur yorick. Evidement il faut avoir la signification de chaque variable pour permettre leur exploitation. La structure de hash-*table* yorick fait tres bien le parallele avec celle du format xml.

Les points principaux a discuter sont:

- obtention de la version de yoga

- gestion de la liste des modeles
- ecriture de fichier oifits
- calcul de donnees synthetiques avec une couverture plus large que celle des donnees observees.
- gestion des erreurs (pour l'instant il existe un niveau tres sommaire qui indique si l'action demandee s'est bien deroulee.)
- probleme de configuration yorick (expliquant des differences de resultats voir meme un disfonctionnement).

4.1 Gestion du code source et distribution

Le code est gere sous cvs (grenoble accede par le compte jmmc):

```
CVSROOT=:ext:jmmc@avae.univ-lyon1.fr:/home/cvsroot/yoga
CVS_RSH=ssh
cvs checkout yoga
```

Voir le GL pour l'aspect distribution.

5 Partie specifique GUI

5.1 Architecture

C'est une application Java autonome qui a ete developpee. Elle peut etre telechargee et executee facilement grace au mecanisme standard Java Web Start sans modification de code.Les details de la distribution et de l'interfacage avec Yoga sont renseignes dans la section 6.

La maquette propose un menu sur la gauche de la fenetre principale. Ce menu est un arbre qui regroupe l'ensemble des elements utiles. Chaque element selectionne dispose d'un panneau specialise ou l'utilisateur peut interagir.

L'utilisation d'xml permet de garantir que les structures manipulees cote client et serveur correspondent aux memes schemas (chaque partie utilise le fichier xsd qui procede aux verifications). Pour l'instant le schema xml ne sait manipuler que des modeles generique. Il ne sait donc pas indiquer un probleme si un parametre manque a une modele de base. L'utilisation conjointe de la librairie de binding castor permet de generer automatiquement le code Java des objects construits dans les schemas.

La presentation des resultats est tres sommaire. Un essai de production en html a ete realise en transformant par feuille de style xsl le xml recu. Le resultat est moins esthetique que des widgets java mais l'exploitation est plus evidentes copier/coller...

L'environnement de developpement netbeans a ete utilise pour faciliter le developpement. Il n'est pas du tout indispensable et genere un code tres bien structure (par contre il faut rester sous l'IDE et eviter certaines manipulations...). Il rajoute juste quelques contraintes sur la gestion du projet de part les quelques fichiers supplementaires.

5.1.1 Fonctionnalites

Chargement des fichiers de donnees observees Des librairies de manipulation de fichiers fits sont utilisees pour permettre a l'utilisateur de charger ses fichiers de donnees.

Visualisation des donnees observees Grace a l'utilisation de topcat http://www.star.bris.ac.uk/~mbt/topcat, certain plot peuvent etre automatise et l'utilisateur a egalement l'acces a toutes les fonctionnalites de topcat.

Construction d'un modele a fitter L'utilisateur peut de maniere graphique construire un ou plusieurs modele compose de briques geometriques simples en relation avec les fichiers de donnees observees.

Partage de parametres Un modele compose peut etre constitue de composants qui partagent un meme parametre. L'utilisation de clic droits dans la table des parametres permet ces ajustements.

Lancement d'un fit Pour l'instant un fit est lance sans le controle de parametre propre au moteur. Un panneau remonte textuellement les actions yoga (avec des informations sur les erreurs). La table des parametres est mise a jour lors de chaque boucle. Le commentaire utilisateur est complete par une information datee permettant de conserver un historique textuel.

Sauvegarde, chargement des fichiers L'utilisateur peut sauvegarder un fit en cours. Le format actuel est en xml. Les fichiers de donnees observees sont integrees dans le fichier oifits en base64.

5.1.2 Configuration systeme necessaire et dependances externes

- Une version de Java superieure a 1.5 doit etre utilisee.
- Le restant des librairies est distribue en meme temps que l'application, ce qui evite a l'utilisateur d'avoir a installer d'autres dependances.
- Par contre cote developpement il faut regrouper l'ensemble de librairies utiles (certaines sont internes mais d'autres proviennent de contributions externes)
- Un acces reseau ou une installation local de yoga et des scripts d'interface sont necessaire.

Liste des librairies utiles au GUI mais exterieures au module mfgui Chacune de ces librairies est sous le repertoire lib du module mfgui

- jmcs : librairie JMMC commune mcs (preferences, gestion du log, affichage des erreurs...)
- topcat : gestion de donnees tabulaires
- jfits : gestion des fichiers fits/oifits
- castor : transformation Objets java <-> document xml
- httpclient: classes de gestion d'echanges HTTP

5.2 Evaluation

5.2.1 Documentation

Documentation utilisateur Pour ce qui conserne l'installation des applications Java sur le poste des utilisateurs il existe une page jmmc dediee aux informations techniques. Les pages de telechargement de l'application modelfitting donnent un pointeur sur cette page.

Pour la documentation sur l'utilisation du GUI, un embrion de documentation est disponible en ligne. Son contenu et son format restent a travailler. Un objectif du GG est de rendre commun aux applications JMMC un navigateur de documentation. Il serait alimente par un fichier hierachise pour chaque application. Ceci permettrait un acces pertinent aux informations avec la possibilite de naviguer par liens hypertextes entre les pages mais aussi avec des boutons d'aides contextuelles dans les panneaux graphiques.

Documentation technique Le code est accompagne de commentaires au format javadoc. Ce memo fait egalement part d'informations. Il faudra egalement fournir la documentation exigee par le JMMC pour des developpements equivalents.

5.2.2 Gestion du code source et distribution

Le code est gere sous cvs:

```
CVSROOT=:ext:cvs.jmmc.fr:/cvs/sw
CVS_RSH=ssh
cvs checkout mfgui
```

La distribution du GUI se fait au travers d'une page web en utilisant le mecanisme Java Web Start.

5.2.3 Gestion des logs

Une gestion locale des log est faite grace a l'API standard java.

L'utilisateur (avance) peut lancer un panneau afin d'ajuster ce qu'il souhaite voir remonte dans le journal. C'est surtout utile pour les developpeurs.

5.2.4 Gestion des erreurs

Un panneau standard s'affiche lorsqu'une erreur (exception au sens Java) est rencontree. Il faudrait rajouter la possibilite de caracteriser l'erreur pour eviter a l'utilisateur de voir le formulaire bug-report s'afficher lorsque l'on rencontre l'exception File Not Found par exemple. Ce panneau affiche la pile d'erreur Java, mais il n'y a pas encore d'association avec un extrait du log courant.

5.2.5 Statistiques

Types de fichiers	nombres de fichiers	nombres de lignes
fichiers Java (developpeur)	23	5000
fichiers Java generes	24	6400
feuilles de transformation xsl	1	40
schema xml	1	280
fichiers html	2	105
script shell	1	130

Le fichier schema est commun au GUI et a la partie d'encapsulation Yoga.

6 Partie encapsultation Yoga

6.1 Architecture

L'architecture proposee se veut la plus simple possible. Elle doit etre egalement le plus souple pour faire fasse a des evolutions.

Le principe est de pouvoir demander avec le maximum de controle possible l'execution de sequences qu'un utilisateur peut demander a yoga (depuis l'interpreteur yorick). La contrainte etant que le GUI est dans un environnement externe a yorick voir meme sur une autre machine. Le reseau doit permettre de lancer ces traitements en rendant l'acces le plus transparent possible.

Suite aux requetes reseaux, le script principal genere:

- un fichier descripteurs
- un fichier batch (invariant)
- les fichiers de donnees observees

Ensuite le batch est lance par yorick. Enfin le travail de yoga est synthetise sous forme d'un fichier xml qui permet la mise a jour du fichier descripteur initiale.

Pour realiser cela il existe a ce jour:

- Une collection de scripts demandant l'execution des fonctions principales (principalement runFit, getYogaVersion, getModelList)
- Un couple de programmes python permettant l'encodage et decodage de fichiers en base64
- des examples de fichiers a fitter
- un ensemble de pages web (html et 1 php)

6.1.1 Fonctionnalites

Fonction de fit Les scripts permettent de faire tourner un fit en ligne de commande. Les resultats sont au meme formalisme mais avec des complements resultants du traitement yoga.

Tests anti-regression En complement des scripts d'utilisation du logiciel, il existe un programme runRegressionTest.sh. Celui-ci boucle sur chaque fichier example et permet d'indiquer les repercutions apres un changement de configuration.

Execution a travers le reseau Une passerelle minime a ete developpee en php. Elle n'est la que pour recevoir les requetes et fichiers associe, puis elle delegue l'execution aux scripts principaux en retournant ce meme fichier xml de synthese.

6.1.2 Configuration systeme necessaire et dependances externes

- Un serveur web avec module php pour faire tourner un "service reseau"
- Une machine *nix avec principalement yorick, yoga, bash, xsltproc

6.2 Evaluation

6.2.1 Documentation

Un readme indique l'utilite de chaque repertoire au niveau principal.

Cette partie n'est pas prioritaire au niveau de la documentation vers l'utilisateur final. Par contre un minimum d'information permettant l'installation sur une autre machine doit etre redige.

Un script env.sh peut etre execute sous l'arborescence des scripts serveurs. Celui-ci tente de lister la configuration necessaire et liste ce qui va ou ne va pas.

6.2.2 Gestion du code source et distribution

Il n'y a pas de gestion de configuration sous CVS actuellement. La distribution n'est pas prioritaire pour l'instant puisque cela correspond a une diffusion vers les equipes qui souhaitent installer un serveur. Cela peut changer completement si l'on souhaite pouvoir utiliser yoga sans le reseau ou son propre yoga modifie.

6.2.3 Gestion des logs

Un niveau tres basique enregistre les executions des scripts principaux. Un fichier present dans l'arborescence log de l'installation conserve l'historique. D'autres informations pourraient etre ajoutees en fonction des besoins.

```
Wed Mar  7 12:15:49 CET 2007 : yoga.sh starts
Wed Mar  7 12:15:49 CET 2007 : runFit.sh starts
Wed Mar  7 12:15:50 CET 2007 : runFit.sh stops
Wed Mar  7 12:15:50 CET 2007 : yoga.sh stops
Wed Mar  7 12:18:16 CET 2007 : yoga.sh starts
Wed Mar  7 12:18:16 CET 2007 : yoga.sh stops
```

6.2.4 Gestion des erreurs

Il n'y a pas trop lieu puisque c'est une partie qui doit etre tres robuste puisqu'elle ne fait que des operations basiques. Cependant si des erreurs sont a remonter (ex file system en ecriture seulement...), celles-ci doivent etre comprehensibles pour le GUI et signale aux administrateurs.

6.2.5 Statistiques

Types de fichiers	nombres de fichiers	nombres de lignes
scripts bash	6	470
scripts python	2	30
feuilles de transformation xsl	2 utiles / 8	350 / 600
schema xml	1	280
fichiers html	2	380
fichier php	1	80

Le fichier schema est commun au GUI et a la partie d'encapsulation Yoga.

7 Conclusion

Les essais de couplage pour piloter yoga depuis un GUI Java n'ont rencontre aucun probleme critique. Les solutions mises en places pour l'interfacage ont ete choisies pour permettre d'avancer rapidement et prouver la faisabilite. Elles pourraient etre conservees mais il serait pertinent de remplacer certains mecanismes par d'autres plus standards. Cela prend un peu plus de temps, mais permet de s'appuyer sur des solutions logicielles plus abouties (on peut gagner en fonctionnalite et en robustesse). Evidement il est necessaire de maitriser des techniques qui imposent un minimum de connaissances techniques (par exemple xml dans une utilisation un peu avancee...). Le point majeur concernerait le remplacement des appels HTTP post par du Web services SOAP.

Il faut par ailleur mettre sous CVS la partie couplage pour pouvoir suivre les evolutions necessaires lorsque la ditribution commencera a se faire.

On pourrait extraire de ce memo un grand nombre d'actions. La prochaine reunion permettra de completer ce paragraphe dans les prochaines versions.

7.1 Points supplementaires

Developpements yorick Le laog porte un effort sur le developpement de modules ou packages yorick en interne. Il peut eventuellement y avoir un partage d'outils, methodes...

Dalia L'obs de Paris Meudon developpe une application VO qui fait du model fitting. Frederic Boone est le principal responsable de dalia http://dalia.obspm.fr/. Meme si le recouvrement entre les domaines scientifiques n'est pas compatible, il est possible d'echanger d'un point de vue techniques sur les approches a avoir. Cela permet d'elargir les techniques mises en oeuvre en ayant plus de support et aussi d'augmenter les retours d'utilisateurs, homogeneiser la presentation des logiciels...

Pages Web Il existe une page wiki a acces restreint. Grenoble souhaiterait mettre dans sa rubrique quelques infos sur le projet auquel il participe comme pour SearchCal http://www-laog.obs.ujf-grenoble.fr/twiki/bin/view/Laog/GRRIL/Informatique/SearchCal par exemple . On peut garder en parallele une page privee si besoin... S'il existe d'autres pages web, autant inclure des references vers les differents sites et eventuellement regrouper d'un cote ou de l'autre....