Rapport de projet : TEP

1.0

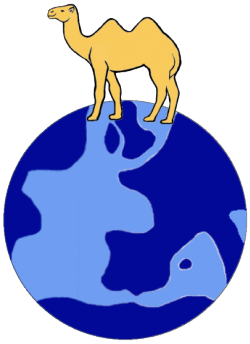
Rapport	**Auteur :** Xiaoyi CHEN	24/11/2006
Objet : Rapport sur le travail réalisé pour le projet TEP	**Destinataire :** Emmanuel Chailloux	**Pages :** 9

Version	Date	Statut	Correcteur	Modifications
1.0	24/11/2006	En Création	-	-

Table des matières

A. Introduction 3
1. Description du sujet et du contexte 3
2. Plan du document 3
B. Corps du rapport 4
1. Rappel et Legacy System 4
2. Test de compatibilité 4
2.1 Préparation de l'environnement de travail 4
2.2 Interprète XUL en O'Caml 4
2.3 Interfaçage XPCOM/O'Caml 5
2.3.1 Le pgcd utilisant XPCOM 5
2.3.2 Calculatrice en XPCOM/O'Caml 6
2.3.3 Extensions Firefox 6
2.4 Synthèse des tests 6
3. Travaux futurs 7
3.1 CDC : Description du projet 7
3.2 CDC : Objectifs fonctionnels à atteindre 7
C. Conclusion 9

A. Introduction



1. Description du sujet et du contexte

Ce document est le rapport du projet de TEP. Le projet en question est le portage du PSTL « O'Xul, interface XUL pour Objective Caml » dans Firefox 2. Pour cela, il nous est demandé de tester la compatibilité des applications réalisées auparavant, avec la nouvelle version de Firefox.

2. Plan du document

Nous allons tout d'abord rappeler le contexte en précisant ce qui a été fait dans le PSTL.
Ensuite les résultats des tests seront donnés, et nous ferons une synthèse sur pourquoi avoir fait ces tests.
Finalement, comme O'Xul est un projet ambitieux, il demande beaucoup de travail. Il faut donc que des personnes prennent la suite. Pour aider ces personnes, un cahier des charges plus précis est donné dans lequel les objectifs sont plus clairement explicités que lors de la phase d'étude.

B. Corps du rapport

Ce projet reprend ce qui a été fait en 2006 pour l'UE PSTL, et est un complément au rapport déjà écrit. Néanmoins, certains points seront rappelés, pour permettre une meilleure compréhension.

1. Rappel et Legacy System

Rappelons ce qu'est XUL et XPCOM :

- XUL est un langage de description d'interfaces graphiques comme les fichiers utilisés avec GNOME.
- XPCOM est un modèle de composants multiplateforme similaire à la technologie COM.

A la fin du PSTL, nous avions un interprète XUL minimaliste qui fonctionnait, avec la prise en charge de la partie JavaScript et des composants graphiques de base de GTK2. Nous avions aussi vu qu'il était possible d'interfacer XPCOM et O'Caml, par l'exemple.
Tout ce qui a été fait, les applications créées, a été développé sous un Linux x86.

De ces applications, nous reprenons d'une part l'interprète réalisé, d'autre part les deux applications basées sur XPCOM et O'Caml, pour réaliser nos tests de compatibilité.

2. Test de compatibilité

2.1 Préparation de l'environnement de travail

Les tests sont réalisés dans un système Linux neuf, c'est-à-dire que la plupart des bibliothèques nécessaires à la compilation et à l'exécution ont été ajouté au fur et à mesure des besoins demandés, à savoir gtk2, lablgtk2, libXt, libIDL, autoconf, etc.

Pour ne pas avoir de problème de fichiers manquants et surtout voir s'il y a des potentiels problèmes de compatibilité de version, nous utilisons une version de Firefox 2 que nous compilons nous-mêmes. On a ainsi le SDK avec les fichiers IDL et les bibliothèques dynamiques. Les options à donner lors de la compilation sont les suivantes :

- mk_add_options MOZ_CO_PROJECT=browser
- ac_add_options –enable-application=browser
- ac_add_options --enable-default-toolkit=gtk2
- ac_add_options --enable-xft

Ces quatre lignes sont à mettre dans le fichier .mozconfig.
Une fois la compilation finie, nous avons une version de Firefox qui nous servira tout le long des tests. De plus nous ne touchons pas aux fichiers du navigateur installé par défaut dans le système.

2.2 Interprète XUL en O'Caml

Tout d'abord, précisons que Firefox 2 intègre une nouvelle version de JavaScript, la 1.7. Pour utiliser la nouvelle version de JavaScript il faut préciser dans le fichier XUL que c'est le cas par l'ajout de la version : <script type="application/javascript; version=1.7"/>.

Notre interprète a été écrit avec utilisation de JavaScript 1.6. La nouvelle version ne fait que rajouter des fonctionnalités, donc il n'y a pas de problèmes avec le code déjà écrit. Le problème se pose à la compilation de SpiderCaml et de l'interprète car elle demande la bibliothèque JavaScript. Comme on veut pouvoir l'utiliser en compatibilité avec Firefox 2, nous allons fournir le fichier *libmozjs.so* de notre Firefox 2 compilé par nous-mêmes. Pour cela, il nous suffit de modifier le fichier Makefile de notre application.

En exécutant la commande suivante :

~/mozilla/dist/bin/run-mozilla.sh parser.exe pgcd.xul

Nous obtenons bien l'affichage de notre application de calcul du pgcd (Figure 2.1). A cause des problèmes de variables d'environnement à fixer, on est obligé de faire appel au script de Firefox.

Figure 2.1 Pgcd par l'interprète

2.3 Interfaçage XPCOM/O'Caml

Cette partie du projet a consisté avant tout à faire une étude de faisabilité. La majeure partie du temps a été de se documenter et de faire des tests basiques. Pour connaitre les résultats de l'étude, se référer au rapport du PSTL. Néanmoins, deux applications simples mais suffisamment complexes ont été conçues pour pouvoir voir apparaitre les problèmes si problèmes il y a.

2.3.1 Le pgcd utilisant XPCOM

Le test de cette application consiste à recompiler l'exécutable et de l'exécuter. La première chose est de vérifier les bons chemins dans le Makefile. Si le chemin d'O'Caml a changé, si le chemin vers les fichiers de bibliothèques Firefox a changé, tout cela doit être modifié en conséquence dans le Makefile. Une fois ces modifications faites, on peut lancer l'exécution, qui se fait par la commande :

~/mozilla/dist/bin/run-mozilla.sh gcd.exe

Le résultat est visible dans la Figure 2.2 :

Figure 2.2 Pgcd par XPCOM/O'Caml

Cette application ne nécessite pas que nous lui fournissons le fichier xul en argument de la ligne de commande, car le choix d'implémentation a été de dire à XPCOM d'aller chercher le fichier exactement dans le répertoire *chrome*, si le fichier n'y est pas, une erreur est lancée. Une autre manière de faire est d'utiliser le protocole *file* sous Linux, au lieu du protocole *chrome*.

Application de base pour l'étude faisabilité, nous avons aussi créé une application plus complexe mais qui utilise le même système des boutons et des champs de texte.

2.3.2 Calculatrice en XPCOM/O'Caml

Tout comme le pgcd, nous modifions le Makefile en conséquence. Nous obtenons au final une application compatible avec la nouvelle version de Firefox 2.

La commande suivante :

```
~/mozilla/dist/bin/run-mozilla.sh calc.exe /chemin_absolu_du_repertoire/calc.xul
```

donne la figure 2.3 :

Figure 2.3 Calculatrice avec XPCOM/O'Caml

Ces tests ont montré que le passage de la version 1.8.0.1 à la version 1.8.1 de Gecko n'a pas été gênant, car les modifications ne sont pas très importantes au niveau de l'API d'XPCOM. Le passage à la version 1.9 va peut être chamboulé tout cela.

2.3.3 Extensions Firefox

Les extensions Firefox utilisent la plupart du temps que des fichiers script que le navigateur analyse. Pour rendre une extension compatible avec une nouvelle version du navigateur, il suffit dans la majorité des cas de changer le fichier install.rdf. Voici ce qu'il faut changer dans le fichier install.rdf d'une extension pour le rendre tout à coup compatible avec Firefox 2 :

```
<em:minVersion>1.5.0.*</em:minVersion>
<em:maxVersion>2.0</em:maxVersion>
```

2.4 Synthèse des tests

Concernant l'interprète XUL, s'il y a des modifications et qu'il se peut qu'il y ait des problèmes de compatibilité avec Firefox ou XULRunner, cela proviendrait de JavaScript, car il utilise JavaScript par l'intermédiaire de SpiderCaml.

Concernant les applications utilisant XPCOM, beaucoup de problèmes de compatibilité peuvent se poser. Les développeurs de Mozilla disent clairement que leur API peut changer radicalement d'une version à l'autre. Mozilla conseille d'utiliser la partie *frozen* de leur API[1], c'est-à-dire celle qui ne change pas peu importe la version. L'utilisation de la partie *unfrozen* par un développeur est à ses risques et périls. Malheureusement, les composants intéressants, puisque XPCOM est un système à composants, font partie de cette partie unfrozen qu'il n'est pas bon d'utiliser sous peine de devoir corriger son application à chaque changement majeur.

Heureusement pour nous, l'application de calculatrice que nous avons conçue utilise la plupart des fonctions frozen, telle que NS_InitXPCOM2 par exemple. La partie unfrozen que nous utilisons est

[1] http://developer.mozilla.org/en/docs/XPCOM_API_Reference

tout ce qui a un rapport avec la partie graphique, c'est-à-dire que si le moteur de rendu change en passant à la version 1.9 (ce qui arrivera avec Firefox 3), il faudra vérifier que les méthodes utilisées existent toujours dans les interfaces XPCOM.

3. Travaux futurs

Le projet en l'état n'est qu'au stade de l'étude et de l'analyse des risques au niveau temps et faisabilité. Néanmoins, elle a permis de conclure qu'il est possible de concevoir des applications O'Caml utilisant certaines technologies Mozilla. Il serait intéressant de passer aux stades suivants. Pour cela, nous allons proposer un cahier des charges (CDC) qui guidera les développeurs qui reprendront la suite du projet.
A noter que ce projet est un projet d'ingénierie. Il reprend tous les concepts de réalisation d'un projet informatique au sens professionnel du terme.

3.1 CDC : Description du projet

L'objectif du projet O'Xul est de fournir une interface graphique à Objective Caml en utilisant les technologies Mozilla, à savoir XUL et XPCOM, qui sont utilisées dans Firefox.

Firefox, le célèbre navigateur de Mozilla devient de plus en plus populaire, que ce soit chez les développeurs ou les utilisateurs. Pour les développeurs, l'utilisation de technologies portables est un atout incontestable. XUL en fait partie. C'est le langage de description des interfaces graphiques. En séparant la partie traitement de l'interface graphique et la description, il permet de mieux gérer les modifications graphiques sans répercussion sur l'application. Il faut donc concevoir un analyseur de fichier XUL et qui crée l'interface graphique correspondante.
Une étude de l'art est déjà faite, et un début d'analyseur existe. Il faut néanmoins l'enrichir de la plupart des composants graphiques XUL. Sous Linux, le choix a été de l'implémenter en GTK2. Sous Windows, il n'y a pas de choix possible puisque le système a son propre implémentation.
Cet analyseur utilise SpiderCaml, l'interprète JavaScript pour O'Caml, ainsi que la bibliothèque d'analyse de balises XML, XML-light.
Cette partie du projet est à réaliser entièrement en O'Caml, et donc ne dépend pas beaucoup des changements de version de Gecko. Le nom du projet vient essentiellement de cette partie.

La deuxième technologie qu'utilise Mozilla s'appelle XPCOM. Elle est le cœur du système car elle touche le plus bas niveau de l'ensemble des technologies. Le but est d'utiliser le moteur de rendu Gecko déjà implémenté par Mozilla, et d'en faire appel. Les applications ainsi réalisées délèguent complètement la partie graphique à XPCOM, et O'Caml ne s'occupe que des traitements dits opérationnels. Cette partie incombe un suivi à chaque changement de version car dépendante du moteur de rendu. Le travail déjà réalisé a permis d'interfacer les deux langages : O'Caml et C++, car Gecko utilise C++. Un composant XPCOM peut être écrit dans différents langages de programmation, mais le C++ est le langage de prédilection. Une application O'Caml utilisant XPCOM existe déjà, et il est recommandé de s'en inspirer pour proposer une bibliothèque de fonctions à O'Caml.

Le travail consiste à reprendre ce qui a été fait et de l'améliorer pour au final, faire en sorte que les deux parties se croisent. C'est-à-dire permettre aux applications O'Caml qui utilise XUL, d'appeler les fonctions bas-niveau XPCOM.

Pour mieux cadrer les développeurs, il est proposé de suivre un certain nombre d'objectifs à atteindre.

3.2 CDC : Objectifs fonctionnels à atteindre

Le but recherché est de proposer une interface graphique riche à O'Caml. D'une part par XUL, d'autre part par XPCOM. De par les choix d'implémentation, il est possible de proposer les deux à la fois.

Les objectifs à atteindre dans la partie XUL sont :

- Ajouter un nombre conséquent de composants graphiques utilisables dans le parseur. Pour arriver à cela, il faut ajouter dans le corps de la fonction de parsing, les widgets souhaités.
- Prise en charge d'appels de fonctions O'Caml. Cet objectif est déjà réalisé, il faut juste voir s'il doit être amélioré.
- Prise en charge de JavaScript 1.7, l'ancienne version étant la 1.6. SpiderCaml est écrit pour du JavaScript 1.6.
- Réaliser une application plus complexe que le PGCD existant.

Pour la partie XPCOM :

- Enrichir le nombre de fonctions utilisables par O'Caml. Ceci est faisable en écrivant des fonctions C++ d'opérations sur le DOM. Le but étant de pouvoir manipuler le DOM du fichier XUL pour accéder à des éléments tels qu'une barre de menus ou des boutons radios, etc.
- Fournir une API générale documentée qui peut être utilisée par n'importe quelle application. Celle existante n'est adaptée que pour l'application pour laquelle elle a été créée.
- La partie graphique n'étant qu'une partie de ce qu'il y a dans Firefox, voir s'il vaut le coup de reprendre « tout » l'API graphique de Gecko.
- Respecter le modèle de composants d'XPCOM en modulant le plus possible.
- Le développeur lambda ne doit avoir qu'à écrire les fichiers XUL pour l'interface graphique et les fichiers O'Caml pour l'application.
- Portage sous Windows de ce qui a été fait. Résoudre les problèmes de compilation potentiels. Les outils nécessaires sont un compilateur C/C++ pour Windows, qui est compris dans Microsoft Visual Studio, Cygwin. Cette partie nécessite de faire une étude complète sur la faisabilité, mais en se basant sur ce qui a déjà été fait, il devrait prendre moins de temps.

Dans les objectifs pour XPCOM, il est demandé de fournir une API, laquelle peut être fournie au développeur O'Caml qui utilise le parseur. Il aura ainsi accès à plus de fonctionnalité, si certaines opérations ne sont pas implantables via le parseur.

C. Conclusion

Ce projet demandait de tester la compatibilité des applications réalisées avec les bibliothèques de la nouvelle version de Firefox. Le problème venant du fait que l'API d'XPCOM peut changer radicalement.

Les tests ont prouvé qu'il n'y avait pas de problèmes de compatibilité. Cela a mis l'accent sur ce qu'étaient les fonctions *frozen*. Comme il n'y avait pas d'utilisation massive de fonctions graphiques dans l'application calculatrice, et que le changement de version n'était pas majeur, il n'y a eu aucun souci. De même la compatibilité ascendante de JavaScript a permis de ne pas affecter la partie sur XUL.

Si problèmes de compatibilité il y a, ce sera probablement pour la version 3 du navigateur de Mozilla. En passant à la version 1.9 de Gecko et en se basant sur XULRunner au lieu d'intégrer directement la partie graphique, il se peut que les applications soient à revoir.