Rochette, Tom (1395966)

Groupe 02

Conception logicielle – LOG2410

Travail personnel

DÉVELOPPEMENT LOGICIEL SUR PLATEFORME LINUX ET WINDOWS

Travail présenté à

Olivier Courchesne

École Polytechnique de Montréal

13 mars 2009

INTRODUCTION

Le domaine informatique a toujours été dominé par trois géants (du côté consommateur) : Linux, Windows et Macintosh. Les trois systèmes étant très différents, cela a toujours causé des complications aux programmeurs qui voulaient rendre leur code accessible à une plus grande population. Ainsi, ils devaient « porter » leurs codes vers une nouvelle plateforme, soit pour passer de Windows à Macintosh ou encore de Linux vers Windows par exemple. Afin de diminuer cette tâche qui se verrait devenir répétitive, ces développeurs ont eut l'idée de regrouper les ensembles de codes similaires afin d'en former des librairies qu'ils pourraient réutiliser pour accélérer le développement d'applications futures.

Avec l'ensemble des librairies qui ont été codés avec le temps (GTK, FTLK, Qt, wxWidgets, OpenGL, SDL, etc.), la majorité des plateformes informatiques ont commencés à devenir de plus en plus uniformes, permettant ainsi de déployer un même code sur plusieurs machines utilisant des systèmes d'opérations différents. On observe de plus la présence d'environnements de développement intégrés (EDI) multi-plateformes tel que Netbeans, Eclipse et Code::Blocks. Toutefois, il reste à déterminer quelle plateforme est la plus apte à permettre un développement qui sera rapide, efficace et de qualité. Afin de débattre de cette question, il est nécessaire d'évaluer certains aspects essentiels à la création d'un logiciel. Un logiciel peut être très facile à programmer mais être de piètre qualité en termes de performance (facilité de développement). Il peut être très long à programmer mais ne rien coûter (coûts reliées au développement). Les outils que les plateformes nous offrent peuvent être flexibles mais nécessiter une grande connaissance de ceux qui les emploient. Pour le développeur qui souhaite vendre son logiciel, il faut aussi connaître les diverses restrictions qui s'appliquent lorsqu'on emploi une librairie. Bref, il s'agit ici d'un ensemble d'aspects que nous aborderons afin de déterminer quelle plateforme entre Windows et Linux convient le mieux à un programmeur.

LINUX VS WINDOWS

FACILITÉ DE DÉVELOPPEMENT

WINDOWS

Windows est une plateforme relativement simple à apprendre. Elle se focus majoritairement sur l'interface utilisateur, donc peu font la connaissance de la console. Windows propose un ensemble varié d'outils de développement allant du simple traitement de texte (Word) pour la création des documents de spécifications jusqu'aux logiciels de programmation complexe tel que Visual Studio en passant par les logiciels UML comme Borland Together. Bien que ces logiciels soient avancés, ils sont toutefois simples à apprendre dû à la nature intuitive de leur interface. Ce sont des logiciels puissants et très simples à la fois. Par contre, ils peuvent faire peur à plus d'un vu le nombre de « boutons » qu'ils offrent, mais cela reste toujours plus avantageux que d'apprendre par cœur des lignes de commandes.

On retrouve des langages de programmation plus spécifique à cette plateforme tel que Visual Basic et C#. On peut aussi employer des langages propriétaires tel que le WLangage inclus dans Windev, un logiciel de développement similaire à Visual Basic mais avec un langage « supposément » plus simple. Il est à noter que ces trois langages sont employés par des EDI qui permettent de rapidement faire du prototypage d'interfaces ce qui s'avère intéressant dans les phases primaires du développement logiciel.

Il est possible de gérer un projet à l'aide de Microsoft Office Project, qui offre des outils pour faire de la gestion de temps à l'aide de diagrammes de Gantt, ou encore de visualiser les coûts des étapes de productions du logiciel. Un logiciel gratuit similaire, Open Workbench, peut être employé. Une bonne gestion d'équipe facilitera grandement le développement d'un logiciel.

La gestion de révision est importante pour la gestion du travail logiciel et donc il faut être en mesure d'accéder facilement à ces outils. Il existe divers systèmes de gestion de

révisions propriétaires, notons ClearCase (de IBM Rational), StarTeam (de Borland) et Visual SourceSafe (de Microsoft). Mentionnons maintenant quelques aspects importants à propos de chacun d'eux. Étant donné l'accès assez limité à ClearCase pour les programmeurs (licence à 4300$), cela rend difficile son apprentissage. Il faudra donc au nouvel employé une période de temps pour apprendre et s'adapter au logiciel (ou recevoir une formation). StarTeam nécessite une base de donnée de type MS SQL ou encore Oracle dont il faudra faire l'achat. Finalement, Visual SourceSafe s'intègre avec Visual Studio pour ainsi faciliter la tâche aux développeurs afin de soumettre des nouvelles révisions. Ainsi, un programmeur qui utilise Visual Studio ne devrait pas être trop dépaysé s'il doit utiliser un outil qui est intégré avec son environnement de travail.

Il ne faut pas oublier qu'il est aussi possible d'accéder à SVN/CVS (systèmes gratuits) via Tortoise SVN/CVS, qui sont des GUI pour Windows. SVN/CVS sont des systèmes centralisés, c'est-à-dire que le code développé et mis dans l'entrepôt se retrouvera sur une unique machine. Pour contrer le problème d'unicité, un nouveau modèle d'entrepôt a été réalisé sous le nom de Bazaar, qui va déployer les révisions sur un ensemble de serveurs (permettant ainsi d'éliminer le problème de serveurs non accessibles).

Pour la maintenance et les tests de programmes, il existe quelques GUI pour remplacer l'affichage à la console des applications tel que NUnit et CppUnit. Ces applications nous donne généralement en sortie l'ensemble des problèmes relatifs à nos tests afin qu'on les archives pour que d'autres puissent ensuite voir les résultats.

Le déploiement d'applications est généralement réalisé à partir d'un logiciel tierce partie qu'il faudra apprendre en lisant la documentation du vendeur (NullSoft scriptable install system, InstallShield, .msi Microsoft Software Installer).

LINUX

Du côté de Linux, un utilisateur doit s'attendre à être plus près de la console, c'est-à-dire d'être en mesure d'en faire un usage régulier. Linux est très puissant lorsqu'on sait se

servir efficacement des outils qui sont mis à notre disposition. Toutefois, pour être en mesure d'acquérir cette puissance de travail, il faudra passer un peu plus de temps que sur Windows et dans des conditions peut-être un peu moins agréables pour certains qui ne sont pas familier avec les concepts relatifs à la ligne de commande.

Par contre, il ne faut pas oublier que Linux offre aussi une gamme d'EDI tel que KDevelop ou Code::Blocks. Ces outils sont très développés, et presque équivalent à Visual Studio. La différence principale est que sous Linux, la majorité vont préférer travailler avec GCC/G++ comme compilateur alors que sur Windows, on emploiera le compilateur intégré avec Visual Studio.

Il est très facile sous les diverses distributions de Linux d'obtenir des outils de développement, et ce en faisant un peu de recherche. Le désavantage est que l'on doit alors trouver une gamme de logiciels qui répondront à nos besoins. Si aucun logiciel ne répond parfaitement à nos besoins, il faudra alors modifier celui qui s'en approche le plus. Avec aptitude (sous Debian) ou yum (sous Fedora/Red-Hat), il nous est facile d'installer/configurer et maintenir les logiciels que l'on a installé sur nos stations de travail. La mise en place d'un environnement de travail n'est donc jamais une tâche bien difficile.

Les tests d'applications sont effectués par les mêmes outils sous Linux (CppUnit), toutefois, si l'on désire une interface graphique, il faudra faire appel à d'autres logiciels que ceux qui viennent avec CppUnit pour obtenir le GUI (étant donné qu'ils sont généralement pour Windows uniquement). On peut donc employer QxRunner, basé sur Qt et donc multi-plateformes.

GNU offre deux outils intéressants qui facilite le déploiement d'applications, soit configure et make. Le premier outil permet au développeur d'obtenir la configuration de l'ordinateur lorsqu'on son application est déployée afin de permettre une compilation plus adapté à la plateforme où son logiciel sera employé. Le deuxième est make, qui est simplement un script contenant un ensemble de lignes de commandes qui s'occupent de faire la compilation, le déplacement et l'appel de divers programmes pour générer

un ou des exécutables. Make install nous permet simplement de déplacer ce qui a été construit par make dans le répertoire approprié (si nécessaire).

COÛTS RELIÉS AU DÉVELOPPEMENT

WINDOWS

Sous Windows, il faut s'attendre à payer un ensemble de produits, à commencer par Windows lui-même. La version actuelle, Vista, coûte environ 300$ par licence. Ensuite, il faut obtenir Visual Studio 2008 Professionnel qui coûte 800$ par copie. Vous pouvez aussi opter pour les produits de la compagnie Borland, tel que C++ Builder, à 875$ la licence.

La documentation sera probablement réalisée avec Word puis transformée sous format PDF ce qui implique l'achat de deux logiciels (ou suites) supplémentaires, soit Microsoft Office, au coût de 700$ la licence et Adobe Acrobat à 450$. Si l'on désire faire la conception de notre produit à l'aide de l'UML, des logiciels tels que Borland Together (~4000$/licence) ou Rational Rose d'IBM (~4000$/licence) peuvent être acquis. Il est aussi possible d'utiliser des alternatives gratuites tel que StarUML.

Les systèmes de révisions mentionnés plus tôt varient grandement dans leur coût. ClearCase peut être acheté au prix de 4400$/licence, StarTeam à 1600$/licence et finalement Visual SourceSafe à 550$/licence.

LINUX

Du côté de Linux, le système d'opération est généralement gratuit. Il est possible d'obtenir des versions « payantes », au sens où vous payez pour un ensemble de paquets (programmes) et du support technique par téléphone (ou courriel) durant une période de temps déterminée. Prenons par exemple Red-Hat qui offre une licence Desktop (de bureau) pour 180$ par année, support technique par interface web.

Pour la création de documents, on peut utiliser Open Office qui est un équivalent à Word. Pour ce qui est de distribuer la documentation en format PDF, on peut utiliser

GhostScript qui convertira un PostScript en PDF, et ce, gratuitement. Il est aussi possible d'écrire la documentation en format LaTeX et de la « compiler » au fur et à mesure, mais cela est plus complexe qu'un simple document Word/Open Office.

Linux offre une gamme d'outils de création de diagrammes UML tel que Dia et Kivio. Kivio est un logiciel simple pour dessiner des formes et de petits diagrammes mais il ne faut pas pousser plus loin. Du côté de Dia, il est possible de réaliser l'ensemble des diagrammes UML requis pour construire un projet complexe.

Finalement, il faudra penser à ajouter des coûts de formation des employés qui sont majoritairement des utilisateurs de la plateforme Windows. Par exemple, SavoirFaireLinux.com[1] offre des formations de 3 jours afin d'apprendre les concepts de bases de Linux (appelé SFL101 - Linux Concepts Fondamentaux). Cette formation coûte 1175$ par personne.

FLEXIBILITÉ DES OUTILS

Il y a un intérêt particulier récemment pour les frameworks/toolkits portables sur l'ensemble des plateformes (Linux/Windows/Mac). On peut nommer par exemple OpenGL, SDL, Qt et GTK+. Ces outils permettent ainsi aux logiciels que l'on développe d'être flexible au sens qu'ils pourront être installés sur plusieurs plateformes différentes, nécessitant uniquement une recompilation du code source.

WINDOWS

Windows offre des outils qui sont généralement robustes et très complets, mais du côté de leur flexibilité, c'est un peu moins intéressant. Puisque les applications sont généralement propriétaire, il faut espérer que nos outils acceptent des plugins. Et encore là, il faut savoir comment ceux-ci fonctionnent et comment en écrire nous-mêmes pour combler nos besoins spécifiques. De plus, un plugin n'est jamais aussi flexible qu'une modification directe dans le code source de l'application.

[1] Voir Références, *Formation Linux*

LINUX

Linux détient un avantage fulgurant dans le domaine de la flexibilité de ses outils. Non seulement la majorité de ceux qui programment des applications sous Linux ont en tête de le rendre le plus portable possible, leur code est généralement entièrement accessible à quiconque (généralement sous des licences tel que GPL, BSD, Apache, X11, etc.). Cela veut donc dire que n'importe qui peut prendre la source et l'adapter à ses besoins spécifiques ce qui confère un grand avantage lorsque certains outils répondent déjà à nos besoins mais qu'il manque seulement quelques détails.

Le seul hic est qu'il faudra vérifier les licences relatives au code que l'on vient d'employer, surtout si l'on compte revendre cette même application. Aussi, faut-il penser connaître les outils que le premier développeur a employés pour écrire son programme, car s'il utilisait GTK et que vous ne connaissez que wxWidgets et que vous implémentez du code additionnel avec cette librairie, votre code deviendra rapidement lourd et difficile à gérer.

Finalement, il ne faudrait pas oublier de mentionner les divers langages pour écrire des scripts qui sont supportés par Linux tel que Perl et Python. Ces langages interprétés permettent de rapidement écrire de petits programmes qui réaliseront des tâches qui pourraient s'avérer longues et pénibles si réalisés manuellement. Il est possible d'obtenir rapidement ces outils en faisant apt-get install perl python ou encore yum install perl python selon la distribution.

EXPÉRIENCE REQUISE DES DÉVELOPPEURS

WINDOWS

Un développeur doit au minimum savoir se servir de ses outils. Sur Windows, ce sera généralement Visual Studio et un outil de contrôle de révision. Afin de développer des logiciels/outils avec interface graphique, il faudra apprendre une partie de l'API de Windows.

Si on regarde des langages plus spécifiques à Windows tel que C# ou Visual Basic, il faut premièrement connaître le langage de base puis ensuite apprendre à se servir des diverses librairies qui viennent avec ce langage.

LINUX

On s'attend à ce qu'un programmeur qui emploi la plateforme Linux soit en mesure d'utiliser la variété des outils que celle-ci offre. Ceci veut donc dire être capable d'utiliser la console ainsi que les divers langages de scriptages tel que Perl ou Python. Cette connaissance est nécessaire lorsqu'il s'agit de rapidement réaliser de courts scripts qui feront une tâche très répétitive de manière quasi instantanée (enfin, en beaucoup moins de temps qu'un être humain). Il faudra aussi que les développeurs aient connaissance des API spécifiques à la plateforme, comme pour Windows.

Un aspect important de Linux est le système de droits et permissions. Linux se base sur un système de groupes et d'utilisateurs. Chaque fichier dans le système de fichiers contient une partie réservé à cette information. Même si ce système est fort simple et surtout efficace, il peut des fois poser des problèmes aux débutants qui n'attribuent pas les permissions correctement à leur fichiers et qui pourront chercher des heures quel est le problème sans le trouver.

Afin de s'éviter les frais de formation, il faudra que nos développeurs soient déjà familier avec l'environnement Linux. Cela veut donc dire être à l'aise avec la console et la variété de services que Linux emploi pour fonctionner.

TYPE D'APPLICATION (COMMERCIALE, CODE LIBRE)

WINDOWS

Windows est une plateforme commerciale et bien peu de logiciels sont développés à code source ouvert, et ceux qui le sont proviennent généralement de codes réalisés sur la plateforme Linux.

Un logiciel sera donc généralement un shareware (essai de 30 jours et par la suite fonctionnalités réduites ou encore impossibilité d'accéder au logiciel) ou tout simplement un logiciel nécessitant une licence.

L'avantage d'acquérir un logiciel payant est que généralement, on peut s'attendre à obtenir un logiciel de meilleure qualité avec une assurance que si des bogues sont découverts, les auteurs du logiciel le mettront à jour. Toutefois, ce n'est pas toujours le cas car certains logiciels sont simplement développés pour que les auteurs y gagnent de l'argent sans vraiment y retoucher dans le futur. Ceci s'applique uniquement aux logiciels de petite envergure.

LINUX

Linux de son côté, est presque entièrement à code source ouvert dû à la nature même de la licence du noyau (GNU GPL 2[2]). Ainsi, les diverses distributions qui en font son utilisation doivent rendre accessible l'ensemble de leurs sources conformément à cette licence. La majeure partie des logiciels qui composent ces distributions sont généralement retrouvables dans ce qu'on appel des entrepôts, qui sont en fait des serveurs contenant l'ensemble des paquets (logiciels) configurés pour la distribution spécifique. Puisqu'un logiciel développé peut se retrouver dans diverses distributions, il faut que son code source soit accessible afin d'en permettre sa recompilation, une des étapes cruciales lors du déploiement d'une distribution Linux sur un poste de travail.

On notera toutefois que Linux est très répandu dans le domaine des applications serveurs puisque l'on considère le système d'exploitation comme étant beaucoup plus stable que Windows. Ainsi, des projets tels qu'Apache, MySQL et PHP ont vu le jour sur ce système d'exploitation. Bien que ces logiciels soient à code source ouvert, il n'en reste pas moins qu'ils sont financés par de grandes entreprises afin que le projet continu à évoluer. On comprendra que ceux qui le financent sont ceux qui l'utilisent dans des buts commerciaux.

[2] Voir Références, *Noyau Linux*

LINUX OU WINDOWS?

Pour cette étude de cas, l'objectif est de déterminer laquelle des deux plateformes est la plus apte à fournir de bons résultats à une entreprise qui déciderait de se lancer dans le développement de jeux vidéo. Que ce soit en terme de réalisation d'un produit de qualité, de facilité à gérer le développement, tester le produit ou encore à le documenter, nous évaluerons les divers aspects que nous avons vu précédemment pour en faire une évaluation finale et décisive sur le système d'exploitation qui semble être le plus approprié pour le développement de jeux vidéos à l'heure actuelle.

WINDOWS

Si l'on compte développer pour les consoles de jeux, et surtout pour le Xbox, la librairie Direct X est un choix justifié. Toutefois, pour pouvoir en faire son utilisation (et la tester), il faut se trouver sur Windows, car elle fait l'utilisation directe des API de Windows et donc ne peut être utilisée ailleurs, sauf sur le Xbox. Cela est donc intéressant puisque vous pourrez donc développer un jeu sur Windows, puis ensuite être en mesure de le porter au Xbox sans trop de difficulté ce qui augmente ainsi votre portabilité. Toutefois, si on préfère porter notre jeu vers d'autre système d'opération, on préfèrera utiliser Open GL qui peut s'employer sur Windows, Mac OS et Linux.

Windows offre de plus XNA, qui est un ensemble d'outils et un framework. Toute la documentation nécessaire pour l'employer est incluse lors du téléchargement de ce que Microsoft appel XNA Game Studio. Il permet ainsi d'intégrer du contenu à un jeu rapidement (ce qui pourrait s'avérer complexe dans d'autres situations, tel que de convertir un modèle d'un logiciel tel que 3dStudioMax en un modèle 3D pour DirectX par exemple). De plus, l'objectif même de cet outil est de permettre aux développeurs d'économiser du temps en n'ayant pas à écrire du code pour des librairies souvent nécessaire dans ce domaine.

Visual Studio est un environnement de développement intégré fort puissant. Il intègre l'ensemble des outils qui nous sont nécessaire pour écrire du code de façon rapide et

efficace. En pesant sur un seul bouton, nous sommes en mesure de demander au compilateur de faire la compilation de notre logiciel. Si une erreur survient, on peut la retracer rapidement à l'aide des outils de débogage qui sont inclus. Son outil IntelliSense nous aide à compléter nos appels de fonction et nous permet aussi de voir quels sont les arguments nécessaires pour appeler la fonction correctement, ce qui nous sauve beaucoup de temps lors de l'écriture.

Puisque la majeure partie de notre audience (la clientèle de notre client) sont des utilisateurs de Windows, il faudra donc que ceux qui développent soient en mesure de tester le code qu'ils créent pour en éliminer le plus de bogues. Pour ce faire, ils devront donc utiliser la plateforme Windows, qui s'avèrera donc une nécessité.

Dans la majorité des développements de jeux à moyennes et grandes échelles, on voudra développer des outils qui permettront à d'autres personnes de créer rapidement du contenu pour le jeu sur lequel on travaille. La majorité des jeux nécessitent la création de cartes qui constituent l'environnement dans lequel le joueur devra se déplacer pour atteindre des objectifs. Toutefois, chaque jeu emploi différentes méthodes pour cartographier les informations du jeu (position du joueur, des ennemis, des murs, des objets, etc.) et donc il faudra créer un outil qui permettra à un concepteur de cartes de rapidement ajouter et modeler sa carte en utilisant un système compatible avec celui du jeu. C# avec le framework .NET répond très bien à ce problème. Étant donné la facilité avec laquelle il permet de créer des interfaces graphiques, il faut très peu de temps pour développer l'outil et y apporter des modifications lorsque cela est nécessaire.

CONCLUSION

Nous terminons à présent notre évaluation de Windows et de Linux en vous rappelant les points observés :

- Windows offre un environnement de travail facile à apprivoiser alors que Linux offre une courbe d'apprentissage un peu plus lente.

- Linux coûte moins cher à déployer que Windows. Toutefois, certains outils de bureautique sont beaucoup plus développés sur Windows.
- Linux est plus flexible en termes de développement, permettant l'accès direct aux sources pour modifier les programmes. Windows, en revanche, nécessite l'implantation de plugins dans le logiciel à l'avance.
- Windows demande moins d'expérience dû à sa forte nature orientée GUI. Linux pourrait demander des heures de formations ainsi que l'apprentissage de certains systèmes spécifiques.
- Windows offre majoritairement des logiciels à code source fermés alors que Linux offre des logiciels à code source ouverts.

Pourquoi suggérons-nous l'emploi de Windows présentement? Car son IDE principal, Visual Studio, est facile à employer et efficace à la fois, parce que bien que le système d'exploitation demande plus de fonds initialement, on y gagne la productivité fournie par les divers logiciels qu'il offre et qui demandent peu d'expérience à ceux qui les emploient ou désirent les employer.

RÉFÉRENCES

APPLICATIONS

IDE

Borland C++ Builder

http://www.programmersparadise.ca/ppi_ca/Product.aspx?skupart=CGI%2037

KDevelop http://www.kdevelop.org/

Microsoft Visual Studio

http://store.microsoft.com/microsoft/Visual-Studio-2008-Professional-Edition-Full/product/5443D1A2

UML

Borland Together http://www.programmersparadise.ca/ppi_ca/Product.aspx?sku=B19%2016101A03

Dia http://live.gnome.org/Dia/

Kivio http://www.koffice.org/kivio/

Rational Rose

http://www-142.ibm.com/software/dre/ecatalog/detail.wss?locale=en_CA&synkey=G114995K68580F30

TESTS

QxRunner http://qxrunner.systest.ch/

TOOLKITS/FRAMEWORKS

GTK http://www.gtk.org/

Librairies de développement de GUI

http://en.wikipedia.org/wiki/List_of_widget_toolkits#Based_on_C_or_C.2B.2B_.28including_bindings_to_other_languages.29

Qt http://www.qtsoftware.com/

DOCUMENTATION

Adobe Acrobat http://www.adobe.com/products/acrobat/

Microsoft Office http://office.microsoft.com/fr-ca/suites/FX101677751036.aspx

GESTION DE RÉVISIONS

Bazaar http://bazaar-vcs.org/

Borland StarTeam http://www.programmersparadise.ca/ppi_ca/Product.aspx?skupart=B19%2005

IBM ClearCase

http://www-142.ibm.com/software/dre/ecatalog/detail.wss?locale=en_CA&synkey=Z012568L96063G61

Microsoft Visual SourceSafe

http://store.microsoft.com/microsoft/Visual-SourceSafe-2005-Full/product/25994351

SVN http://subversion.tigris.org/

LINUX

Formation Linux http://formation.savoirfairelinux.com/fr/cours.php?id=SFL101

Noyau Linux http://fr.wikipedia.org/wiki/Noyau_Linux

Red-Hat http://www.redhat.com/

AUTRE

GCC/G++ http://gcc.gnu.org/

Licences des logiciels http://fr.wikipedia.org/wiki/Licence_de_logiciel

Microsoft XNA http://www.xna.com/