La fin de l'enfer des « DLLs »

UN OUTIL DE SUPPORT AU DÉVELOPPEMENT

Par: Nelson Ruest et Danielle Ruest
Entreprises Résolution
www.reso-net.com

Cet article est le troisième d'une série sur le support au développement avec les technologies Windows de Microsoft.

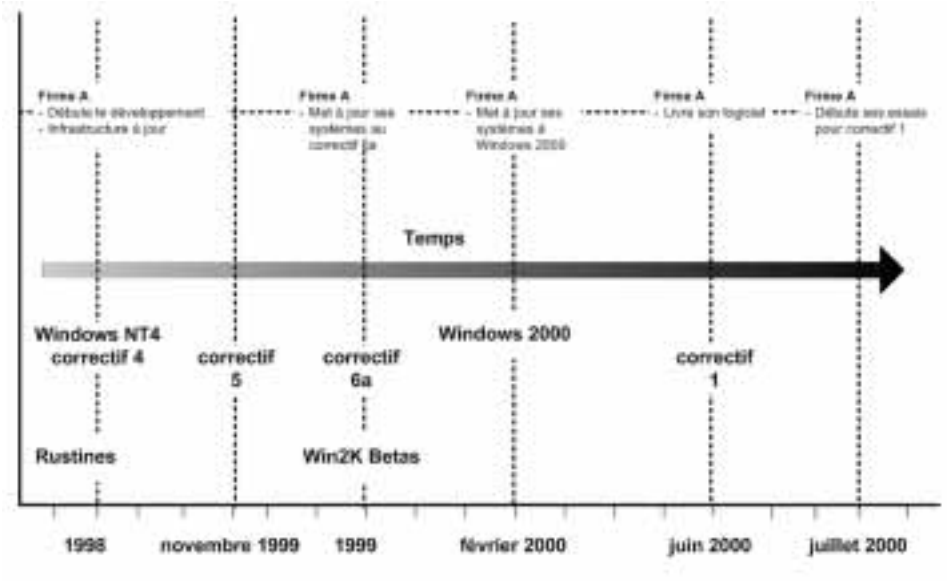
Tout programmeur Windows connaît bien le concept de DLL. Tout programmeur Windows a sûrement même déjà vécu « l'enfer des DLLs ». C'est le problème le plus connu dans le marché du développement pour Windows. C'est en fait, le problème qui est le fil conducteur de plusieurs innovations par Microsoft, notamment dans Windows 2000 et Windows XP. C'est aussi le fil conducteur de la nouvelle initiative du cadre de développement .NET (.NET Framework) qui fait maintenant fureur chez Microsoft. Mais, l'enfer des DLLs, qu'est-ce que c'est vraiment?

« Les DLLs (« Dynamic Link Libraries », ou « bibliothèques de liens dynamiques ») sont des éléments essentiels de Windows. Un DLL contient des fonctions qui sont appelées par un programme exécutable au cours de son exécution. En d'autres termes, un DLL est une bibliothèque de fonctions avec lesquelles un programme peut se lier dynamiquement, » nous dit le site Web de Microsoft France.

En effet, les DLLs sont des composants réutilisables qui sont à la base du système de programmation de Windows. Ils servent à plusieurs fonctions. Par exemple, le système Windows lui-même est un système composé de DLLs. Il sert à offrir à tous les programmes qui s'y exécutent une série de fonctions préprogrammées. Celles-ci incluent la capacité de parler à une imprimante, à un écran, d'accepter des données à partir de claviers, de souris, de communiquer sur réseau, etc. Ainsi, le programmeur Windows n'a pas à se soucier de ces fonctions lors de la création d'un programme. (Voir « Le défi des environnements distribués », InfoQuébec, volume 24, no. 8.)

Un programme Windows est comme un chef d'orchestre ou un maestro qui ne fait que diriger les actions des membres de l'orchestre ou dans ce cas, des DLLs. Cette approche a de grands avantages. Puisque le programme est un chef d'orchestre, il ne charge pas toutes les fonctions en mémoire lors de son démarrage. Microsoft Word, par exemple, ne fait que démarrer son environnement de fonctionnement lorsqu'on le fait partir. Ensuite, au fur et à mesure que l'utilisateur se sert de fonctions différentes, Word fait appel à chaque fonction, la chargeant et la retirant de la mémoire vive de l'ordinateur au besoin.

Cette approche simplifie aussi la programmation en la divisant en petits composants. Ainsi le programmeur n'a qu'à se concentrer sur la fonction requise durant sa préparation. Cette méthode de programmation a fait bien des heureux car elle supporte pleinement le modèle distribué de programmation. Mais, elle a aussi fait bien des malheureux.



L'horaire de production a de la difficulté à suivre la mise à jour des infrastructures.

L'enfer des DLLs

Eh oui, le problème du modèle des DLLs est le modèle lui-même et la nature humaine qui l'utilise. Prenons Word encore une fois comme exemple. Microsoft programme une nouvelle version de ce logiciel. Mais, puisque ce logiciel réside sur Windows, Microsoft n'a pas à programmer toutes les fonctions de Windows que Word utilisera. Alors, le programme Word, ou le maestro de Word, se servira autant de DLLs privés — ceux qui sont particuliers à Word et qui sont livrés avec le produit — que de DLLs publics — ceux qui sont particuliers à Windows.

Le problème ou l'enfer des DLLs provient du fait que lors de la préparation et de la livraison de leurs programmes sur le marché, les manufacturiers de programmes Windows ont pris l'habitude de créer un assemblage de DLLs autant privés que publics. C'est compréhensible, si un manufacturier crée un programme et que celui-ci est testé et re-testé avec un gabarit précis de DLLs publics (une version particulière de Windows), ce dernier voudra s'éviter des problèmes en incluant les composants publics requis avec le programme.

Alors, lors de l'installation du programme sur votre ordinateur, celui-ci recopiera des composants qui sont fort probablement déjà sur votre système, histoire d'éviter des appels de support au manufacturier. La logique est bonne du point de vue du manufacturier. « Nous savons que notre produit fonctionne avec ces composants, nous devons donc les inclure avec notre produit. »

Elle est moins bonne de votre point de vue car il est fort probable que si vous êtes un bon utilisateur de Windows, vous gardez toujours votre système à jour avec des correctifs et des

ajouts rendus disponibles par Microsoft. Si c'est le cas, le composant qui se trouve sur votre poste est fort probablement plus récent que celui livré avec le logiciel que vous venez de vous procurer et lors de l'installation de celui-ci, votre version du composant sera détruit!

Ça c'est l'enfer des DLLs. Un système fonctionnel est rendu instable car les manufacturiers de logiciels s'acharnent à livrer des composants de Windows avec leurs produits. Le résultat : un système instable et le fameux écran bleu de Windows!

Microsoft a fait bien du chemin pour éviter ce problème. Windows 2000 et Windows XP incluent plusieurs fonctions qui évitent cette situation. Les nouvelles innovations de Windows XP sont décrites d'ailleurs dans « La gestion du système local » (InfoQuébec, volume 26, no. 6).

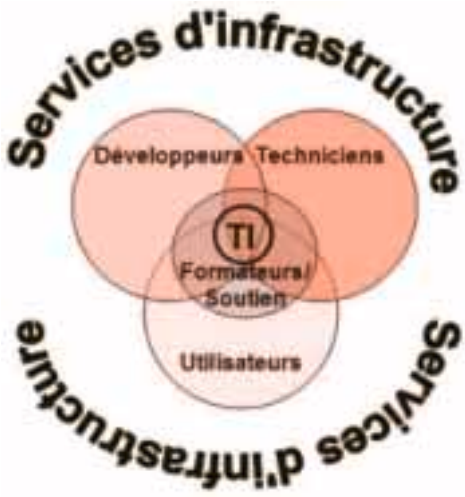
Mais pourquoi ce problème existe-t-il? D'où vient cette crainte des manufacturiers? Pourquoi doivent-ils inclure des composants Windows avec leurs logiciels? Est-ce seulement une question de nature humaine?

Un problème d'horaire de production

Prenons par exemple, la firme A qui développe un nouveau logiciel pour son utilisation interne. Cette firme évalue l'horaire de développement à une période de 18 mois, alors elle se tourne vers son service de gestion d'infrastructure et demande à celui-ci quel gabarit d'outils elle devrait utiliser pour développer. Elle débute ainsi son développement avec un gabarit incluant des outils tels Windows NT avec correctif numéro 4 et Office 97.

Or, lors du cycle de développement qui procède à grands pas, le groupe d'infrastructure continue à faire évoluer la plateforme Windows. Durant une période de 18 mois, il n'est pas impossible de voir une série de mises à jour qui peuvent inclure l'application de correctifs et même une mise à jour du système d'exploitation vers Windows 2000 ou XP.

Alors, qu'arrive-t-il à nos programmeurs? Ceux-ci ont démarré avec un certain gabarit d'outils qui devient rapidement désuet. Ils essaient de suivre en mettant leurs composants à jour en même temps que les gens d'infrastructure, mais quelle que soit l'envergure de leurs efforts dans ce domaine, ils seront toujours désuets. Pourquoi? À cause du cycle de développement. Avant de livrer leur application, ces développeurs doivent effectuer une batterie d'essais: unitaires, fonctionnels, intégrés, acceptation, pseudo-production. Dépendant de l'envergure du programme à livrer, ces essais peuvent prendre jusqu'à plusieurs mois. Pire, il leur est impossible de mettre à jour leur infrastructure de programmation durant ces essais car il est impossible de programmer sur une infrastructure en mouvement. Ils livrent donc un logiciel qui fonctionne sur une infrastructure désuète qui n'est plus en vigueur.

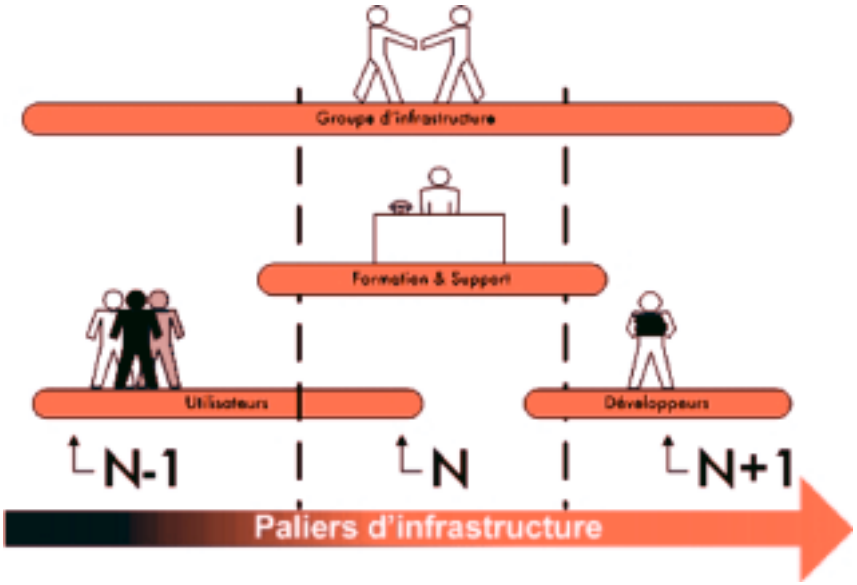


Le groupe d'infrastructure doit supporter les besoins de plusieurs clientèles.

La fin de l'enfer des DLLs

Comment éviter ce problème à tout jamais? Ce n'est pas facile, mais c'est entièrement possible. La première partie de la solution réside avec le groupe d'infrastructure. Il arrive trop souvent que ce groupe ne considère pas les besoins particuliers de toutes ses clientèles, surtout celles de développement. Une infrastructure technologique doit supporter plusieurs clientèles :

- Les utilisateurs — qui sont la clientèle la plus évidente
- Les formateurs — qui doivent avoir accès à tous les environnements d'infrastructure car ils forment à tous les niveaux



Les paliers d'infrastructure doivent rencontrer les besoins des différentes clientèles.

- Le groupe de soutien — qui doit avoir la même chose que les utilisateurs plus des outils de dépannage
- Le groupe d'infrastructure — qui doit avoir une infrastructure supportant l'administration des infrastructures
- Les développeurs — qui selon leur projet de développement, court ou long terme, doivent avoir accès, non pas à l'infrastructure courante, mais plutôt à l'infrastructure à venir du parc informatique

Deuxièmement, la gestion des infrastructures doit viser des paliers différents par clientèle. Il arrive souvent que l'on retrouve dans un parc informatique au moins deux paliers d'infrastructure : le palier courant et le nouveau palier qui est en voie de déploiement. Le palier courant est considéré comme le palier « n moins 1 ». Le palier de remplacement est « n ».

Mais, si on veut éviter l'enfer des DLLs ou l'ajout de DLLs désuets provenant des environnements de développement dans notre parc informatique, il faut instituer un troisième palier : « n plus 1 ». Ainsi, le développeur ne fonctionne pas avec le palier courant, mais plutôt avec le palier à venir. Cette approche demande souvent d'effectuer de longs projets de développement avec de toutes nouvelles technologies, même peut-être des technologies en période de conception (dites « bêta ») qui ne sont pas encore disponibles commercialement.

Il s'agit, en fait, de décaler les paliers d'infrastructure et de s'assurer que nos environnements de développement sont toujours à l'avant des paliers de production. Alors, lors de la livraison des applications développées, il y aura une synchronisation de paliers et l'application peut être livrée en même temps qu'un rehaussement de palier de production.

Il n'y a rien de plus frustrant que de découvrir que notre organisation vient d'investir plusieurs mois de développement dans un produit qui est basé sur une infrastructure désuète et qui demande une refonte totale afin de modifier cette infrastructure. Par exemple, un produit qui est basé sur Microsoft Office 2000 alors que Office XP en est déjà à son premier correctif, mais, puisque l'application est basée sur Office 2000, l'organisation doit retarder son acceptation d'Office XP et attendre une refonte du système.

Un futur encourageant

Une solution complète de support au développement doit inclure plusieurs volets différents. Un de ces volets est l'infrastructure de développement. Ces besoins seront bientôt choses du passé avec l'arrivée du cadre de développement .NET de Microsoft car celui-ci est indépendant de plate-forme. Cette discussion est la base de notre prochain article.