

**Microsoft**



## **Interopérabilité**

*Un guide à l'usage des débutants pour  
comprendre les enjeux de l'interopérabilité.*

Version : 2.2

Dernière mise à jour : Février 2008

Auteur : Lorenzo Madrid

Les informations contenues dans ce document représentent l'opinion actuelle de Microsoft Corporation sur les points cités à la date de publication. Microsoft s'adapte aux conditions fluctuantes du marché et cette opinion ne doit pas être interprétée comme un engagement de la part de Microsoft ; de plus, Microsoft ne peut pas garantir la véracité de toute information présentée après la date de publication.

Ce livre blanc est fourni uniquement à titre indicatif. MICROSOFT N'APPORTE AUCUNE GARANTIE, EXPRESSE, IMPLICITE OU STATUTAIRE, PAR CE DOCUMENT.

L'utilisateur est tenu de respecter la réglementation relative aux droits d'auteur, applicable dans son pays. Aucune partie de ce document ne peut être reproduite, stockée ou introduite dans un système de restitution, ou transmise à quelque fin ou par quelque moyen que ce soit (électronique, mécanique, photocopie, enregistrement ou autre) sans la permission expresse et écrite de Microsoft Corporation.

Microsoft peut détenir des brevets, avoir déposé des demandes d'enregistrement de brevets ou être titulaire de marques, droits d'auteur ou autres droits de propriété intellectuelle portant sur tout ou partie des éléments qui font l'objet du présent document. Sauf stipulation expresse contraire d'un contrat de licence écrit de Microsoft, la fourniture de ce document n'a pas pour effet de vous concéder une licence sur ces brevets, marques, droits d'auteur ou autres droits de propriété intellectuelle.

Sauf stipulation contraire, les entreprises, les produits, les noms de domaine, les adresses électroniques, les logos, les personnes, les lieux et les événements utilisés dans ce document sont fictifs. Toute ressemblance avec des entreprises, noms d'entreprise, produits, noms de domaine, adresses électroniques, logos, personnes ou événements réels serait purement fortuite et involontaire.

© 2006 Microsoft Corporation. Tous droits réservés.

Microsoft, Active Directory, BizTalk, Win32, Windows, Windows Server et Windows Server System sont soit des marques déposées, soit des marques commerciales de Microsoft Corporation aux États-Unis et/ou dans d'autres pays.

Toutes les autres marques déposées appartiennent à leurs propriétaires respectifs.

## **Sommaire**

Introduction .....	4
Portabilité .....	5
Interopérabilité et communication entre systèmes .....	7
Interopérabilité – Un exemple simple .....	9
Interopérabilité et passerelles : Étendre les possibilités .....	10
Interopérabilité et convergence .....	13
SOA et services WEB .....	14
Un standard ouvert n'est pas « open source » .....	16
Standards et interopérabilité : un grand mariage .....	17
Conclusion.....	18
Glossaire.....	20
Bibliographie .....	21

## **Introduction**

---

L'objectif de ce livre blanc est de présenter ce qu'est l'interopérabilité, définir quelques concepts de base et rappeler l'évolution de l'interopérabilité depuis les débuts de l'informatique. Nous présenterons aussi quelques cas dans lesquels l'interopérabilité a joué un rôle majeur pour favoriser la réussite d'initiatives dans le domaine de l'administration électronique (e-gouvernement). Nous en déduirons plusieurs recommandations élémentaires.

Un autre objectif de ce document est de clarifier la différence entre la portabilité et l'interopérabilité selon Microsoft, et ainsi de servir de référence pour une bonne compréhension de ces deux concepts.

Pour une analyse approfondie de l'interopérabilité, nous recommandons « Government Interoperability – Enabling the Delivery of E-Services », un livre blanc Microsoft rédigé par Jerry Fishenden, Oliver Bell et Alan Grose.

En informatique, comme dans beaucoup d'autres sciences, de nouveaux termes sont créés en permanence pour désigner des problèmes techniques. Plus tard, ces mots sont utilisés dans des sens différents par une population plus large, en perdant de vue la signification originelle du terme. Parfois, un mot est utilisé de façon incorrecte, ce qui conduit à des prises de décision erronées. Par conséquence, cette erreur d'interprétation peut influencer de façon négative sur les futurs processus opérationnels des départements informatiques et provoquer des problèmes à l'échelle de l'entreprise toute entière.

Aujourd'hui, les plateformes informatiques se résument à quelques fournisseurs importants, par exemple un site central IBM, des systèmes Linux et UNIX, et Microsoft Windows. Le système d'exploitation Apple représente une petite part du marché dans son ensemble. De nombreux systèmes qui jouaient un rôle important il y a dix ou vingt ans, comme MCP de Burroughs, NOS et SCOPE de Control Data, VAX/VMX et PDP 11 de Digital, ont quasiment disparu. Par le passé, en raison de la multitude de plateformes logicielles et matérielles qui existaient alors, il était nécessaire de résoudre deux besoins importants pour minimiser les problèmes liés à des environnements aussi hétérogènes (la Tour de Babel informatique) : Il s'agissait :

- a) De faire fonctionner la même application sur des plateformes différentes.
- b) De faire parler ces systèmes entre eux, en échangeant des données et des messages.

Une nouvelle terminologie a été créée pour désigner ces deux catégories de problèmes :

- a) La portabilité des programmes.
- b) L'interopérabilité, c'est-à-dire la communication entre systèmes.

Ce livre blanc décrit ces deux concepts en détail, avec des exemples techniques du monde réel.

## Portabilité

---

Au début de l'informatique, les systèmes d'exploitation et les programmes étaient développés pour une plateforme spécifique, en utilisant des langages de programmation qui n'étaient pas totalement compatibles. Chaque fournisseur de matériel possédait son propre langage assembleur ainsi que sa propre implémentation de FORTRAN et de COBOL, les deux langages prédominants et « standards » à cette époque. Les outils de chaque fournisseur présentaient des fonctionnalités « uniques ». Il en résultait que les programmes ne pouvaient pas facilement être transférés d'une plateforme à une autre sans nécessiter de profonds remaniements du code. Pour compliquer le problème, de grandes différences existaient entre les systèmes d'exploitation, et la plupart des matériels ne pouvaient exécuter qu'un seul système d'exploitation, celui de son fabricant. Les programmeurs devaient souvent employer des commandes à bas niveau pour parler à certaines fonctions spécifiques, en plus d'utiliser des formats de fichiers étroitement liés à un système d'exploitation.

Le portage d'une application d'une plateforme à une autre devenait vite un cauchemar, même lorsqu'il s'agissait de passer d'un système d'exploitation à un autre assez proche, comme de VSE à MVS chez IBM, de PDP11 à VAX chez Digital, ou d'un UNIX à un autre. La tâche de passer un programme d'une plateforme à une autre se nomme portage. On la rencontre encore aujourd'hui avec les environnements UNIX.

Au fil du temps, l'adoption de standards dans le domaine de la programmation, comme C, C++ et C# entre autres, ainsi que l'adoption de cadres de développement (les « frameworks ») comme J2EE et .NET, ont réduit les besoins de portage d'une plateforme à une autre. Toutefois, le portage existe toujours lorsqu'il s'agit de passer une application d'UNIX à MVS ou d'UNIX à Windows.

Le concept initial de la portabilité a été défini comme le besoin d'exécuter la même application sur un matériel différent de celui qui constituait la cible d'origine. Par conséquent, la portabilité était essentiellement liée à des contraintes matérielles. Le système d'exploitation était en effet étroitement lié au matériel.

Passons en revue quelques définitions de la portabilité :

Microsoft Encarta définit la portabilité ainsi :

*Programme facilement converti pour s'exécuter sur différents systèmes d'exploitation d'ordinateurs.*

<http://whatis.techtarget.com/> la définit ainsi :

*La portabilité est une caractéristique attribuée à un programme informatique s'il peut être utilisé dans un système d'exploitation différent de celui pour lequel il a été créé, sans nécessiter un travail important. Le portage est la tâche correspondant au travail nécessaire pour que le programme puisse s'exécuter dans le nouvel environnement. En général, les programmes qui respectent les interfaces de programmation standards comme les interfaces du langage C standard défini par X/Open UNIX 95, sont portables. Idéalement, un tel programme doit seulement être recompilé pour le système d'exploitation cible. Cependant, les programmeurs utilisent parfois, en plus des interfaces standards, des extensions du système d'exploitation ou des fonctionnalités particulières qui n'existent pas dans le système d'exploitation cible. Il faut alors supprimer l'usage de ces extensions ou les remplacer par des fonctions comparables du nouveau système d'exploitation. En plus des différences de langage, le portage peut aussi nécessiter des conversions de données et une adaptation à de nouvelles procédures système pour que l'application fonctionne correctement.*

<http://www.techweb.com/> définit ainsi la portabilité :

*La portabilité fait référence à un logiciel qui peut facilement être déplacé d'un type de machine à un autre. Cela implique un produit qui existe sous plusieurs versions, une pour chaque plateforme matérielle, ou qui est capable par sa structure interne de passer de l'une à l'autre. Toutefois, un programme qui peut facilement être converti pour passer d'un type de machine à un autre est aussi considéré comme portable.*

Wikipedia ([www.wikipedia.com](http://www.wikipedia.com)) propose cette définition :

*Le portage est l'adaptation d'un logiciel de telle sorte qu'il fonctionne dans un environnement informatique différent de celui pour lequel il a été conçu à l'origine. Le portage est généralement*

*requis en raison de différences dans les processeurs ou dans les interfaces des systèmes d'exploitation, de l'existence de matériels différents, ou en raison d'incompatibilités subtiles avec le langage de programmation utilisé sur le système cible (voire même l'absence complète du langage de programmation source sur le système cible).*

*La portabilité fait généralement référence à un des deux concepts suivants : Le premier fait référence à la capacité de compiler le code une seule fois (ce qui généralement donne naissance à un code intermédiaire qui est ensuite compilé juste à temps lors de l'exécution), puis de l'exécuter sur différentes plateformes sans modification du code. Le second concept fait référence à la propriété d'un logiciel qui est facile à porter. À mesure que les systèmes d'exploitation, les langages et les techniques de programmation évoluent, le logiciel est porté de plus en plus facilement d'un environnement à un autre. Par exemple, un des objectifs du langage de programmation C et de la bibliothèque standard C consistait à simplifier le portage des logiciels en fournissant une interface de programmation d'application (API) commune à différents matériels.*

*Généralement, le fait d'utiliser des appels de fonctions de haut niveau à la place d'appels de fonctions de bas niveau améliore la portabilité.*

*Des standards internationaux, comme ceux promulgués par l'ISO, facilitent le portage car ils spécifient les détails d'un environnement informatique qui varient peu d'une plateforme à une autre. Souvent, le portage d'un logiciel entre deux plateformes qui mettent en œuvre le même standard (par exemple POSIX 1) revient simplement à recompiler le programme sur la nouvelle plateforme.*

*Il existe aussi un nombre croissant d'outils qui facilitent le portage, comme GCC qui propose plusieurs langages de programmation cohérents entre diverses plateformes, et autoconf, qui automatise la détection de variations mineures dans l'environnement et adapte le logiciel avant la compilation.*

*Deux activités sont en relation avec le portage, tout en restant distinctes de celui-ci : l'émulation et la cross-compilation.*

*Le portage est aussi le terme utilisé lorsqu'un jeu informatique, conçu pour fonctionner sur une plateforme (une console de jeu ou un ordinateur) est converti afin de fonctionner sur une autre. Les premiers portages de jeux vidéo n'étaient pas de véritables portages mais des réécritures complètes. Aujourd'hui, de plus en plus de jeux vidéo sont développés avec des plateformes de développement qui peuvent au final produire un code pour PC aussi bien que pour les consoles les plus répandues. De nombreux portages ont produit des résultats de mauvaise qualité en raison des différences importantes qui ont existé entre les matériels des PC et des consoles de jeux.*

L'Encyclopaedia Britannica définit ainsi la portabilité :

*Utilisable sur de nombreux ordinateurs sans modification (logiciel portable).*

Au vu de toutes ces définitions, il apparaît que la portabilité est essentiellement définie comme la capacité d'utiliser le même programme sur différents ordinateurs sans modification. Cela implique que la plateforme Microsoft Windows offre la meilleure portabilité par rapport aux autres plateformes car n'importe quelle application Windows peut fonctionner sur les systèmes de centaines de fournisseurs différents sans modification.

En réalité, le besoin de portabilité existe uniquement lorsqu'il existe un impératif de faire migrer ou de convertir un logiciel d'une plateforme vers une autre totalement différente (systèmes d'exploitation différents). La portabilité ne doit pas être confondue avec l'interopérabilité. Deux systèmes peuvent fonctionner en harmonie sans que l'application de l'un doive migrer vers l'autre.

## **Interopérabilité et communication entre systèmes**

---

Alors que la portabilité désigne la migration d'une application d'une plateforme vers une autre, l'interopérabilité consiste à permettre à des applications, des plateformes, des systèmes ou des composants différents de se connecter et d'échanger des données entre eux, en clair « de se parler ».

Il existe de nombreux moyens pour assurer l'interopérabilité : (1) Développer un logiciel « interopérable par conception » (par exemple, l'inclusion d'une technologie ou d'une fonctionnalité dans le logiciel qui facilite l'échange de données entre différentes applications, ou la création d'un « traducteur » ou d'une « passerelle » qui s'exécute sur une plateforme et établit des communications avec une autre). (2) Publier et licencier des technologies propriétaires et la propriété intellectuelle concernée. (3) Établir des collaborations spécifiques avec des partenaires, des concurrents, des clients et des gouvernements. (4) Mettre en œuvre des standards (y compris des standards ouverts et des standards propriétaires rendus accessibles) dans des produits et des services.

Chacune de ces voies est importante pour permettre l'interopérabilité, et efficace dans certains contextes. Ce document les abordera toutes sur certains angles mais notre préoccupation première est de montrer comment les standards du marché et les passerelles servent à faciliter l'interopérabilité dans l'informatique.

Pour résoudre les problèmes d'échanges de communications, de nombreux standards comme ASCII, BCD et EBCDIC ont été développés pour harmoniser le stockage des données dans les ordinateurs. Des protocoles de télécommunication, comme Poll Select, BSC1, BSC3 et SDLC, et des protocoles réseau, comme SNA, DECNET, ISO/OSI et TCP/IP, ont été développés pour faciliter les communications entre systèmes.

L'existence de ces protocoles et de ces standards a d'abord permis la connexion entre les ordinateurs et les imprimantes. Plus tard, les ordinateurs ont échangé des données via des bandes magnétiques. Puis sont apparus les échanges directs d'ordinateur à ordinateur et les communications en temps réel.

Dans les premiers temps, ces échanges se limitaient à des données ; chaque application définissait son propre format d'enregistrement. Pour échanger des données entre des systèmes, il fallait donc écrire du code spécifique pour traduire le format d'une application dans le format d'une autre.

Cela conduisait à travailler de façon particulièrement inefficace. En raison de la complexité des systèmes, de leurs besoins spécifiques et du nombre croissant d'applications, les tâches de programmation devenaient très lourdes pour assurer la maintenance du code chaque fois qu'un format de données changeait dans une application. Des milliers de lignes de code étaient nécessaires pour adapter tous les programmes qui exploitaient ce type de format. Pour résoudre ce problème, il a fallu définir des niveaux de protocoles supplémentaires, non seulement pour harmoniser les éléments des données mais aussi les formats des données.

Il a fallu aussi décrire les formats des enregistrements des données et les stocker indépendamment de la logique du programme. Ce simple besoin a conduit à la création des systèmes de gestion de base de données (SGBD) dont les premiers leaders ont été IMS/DB et DL/1 d'IBM, et TOTAL de Cincom Systems. L'apparition des bases de données a constitué une étape importante pour permettre aux programmes et aux systèmes d'échanger des informations dès lors que les données étaient stockées et décrites dans un emplacement unique. Si une modification était apportée à un élément de données, seuls les programmes et les systèmes concernés par cet élément devaient être modifiés. Les communications s'établissaient facilement entre les programmes et les systèmes qui exploitaient la même base de données. De plus, TOTAL était disponible auprès de plusieurs fournisseurs de matériels, simplifiant ainsi le portage de programmes d'un fournisseur à un autre, la base de données restant l'unique véhicule pour transporter les données. Il devenait possible de passer des données d'une application à une autre sur le même ordinateur, ou d'un ordinateur à un autre, via des mécanismes non temps réel comme des bandes magnétiques pour assurer le transport des données.

Bien sûr, il est vite apparu que l'idéal serait de pouvoir échanger des données en temps réel dans les deux sens entre différents ordinateurs, c'est-à-dire de bénéficier d'une interopérabilité plus efficace. De nombreux protocoles de communication existaient déjà, mais il fallait définir une spécification pour le format des enregistrements afin de faciliter les échanges des données entre applications différentes, comme des transactions bancaires, des marchés boursiers, la fabrication ou les télécommunications. Quelques protocoles, conçus en fonction de besoins métier spécifiques, apparurent pour prendre en charge les échanges de

données électroniques (l'EDI – Electronic Data Interchange). D'autres formats de messages pionniers ont rencontré un immense succès, comme ISO 8583 et SWIFT qui interconnectent les institutions financières et les opérateurs de réseau EFT (transferts électroniques de fonds).

Le besoin croissant de mettre au point des systèmes capables d'interopérer automatiquement, la réduction des coûts des télécommunications, et l'existence de protocoles réseau fiables et bien définis ont été les catalyseurs pour une large adoption d'Internet comme système universel capable d'assurer l'interopérabilité entre systèmes. En effet, il est devenu possible d'avoir des systèmes capables de parler à d'autres via des mécanismes d'échange de données externes, mais aussi capables de communiquer entre eux en temps réel. Au lieu d'échanger des fichiers de données entiers entre systèmes, il est devenu possible de transmettre des éléments spécifiques d'informations d'un système à un autre et de recevoir une réponse. À nouveau, l'échange d'éléments de données entre systèmes hétérogènes a nécessité le développement de protocoles standards afin que n'importe quel système puisse comprendre ces données. Cela a conduit à une informatique universelle et omniprésente.

Comme nous l'avons vu, le besoin de résoudre les problèmes d'interopérabilité entre programmes et systèmes a conduit au développement des protocoles. Leur large acceptation en a fait des standards du marché. Les protocoles permettent aujourd'hui aux éléments de données d'être indépendants de la logique des programmes. Grâce à eux, différents systèmes sont capables de comprendre les données échangées, indépendamment de la plateforme qui émet ou reçoit ces données.

Ce qui n'était au départ qu'un problème d'échange de données s'est ainsi transformé en besoin de communication entre programmes puis entre systèmes. Pour assurer l'interopérabilité entre programmes, plusieurs solutions techniques ont été proposées puis adoptées, comme les protocoles fondés sur les messages (RPC), les initiatives de systèmes ouverts et basées sur des objets comme ODBC dans les années 80, les échanges de composants (dans les années 90), et finalement les échanges de messages XML (depuis 2000).

Aujourd'hui, XML est le langage le plus adopté pour répondre aux besoins d'échanges de données et assurer l'interopérabilité entre systèmes et programmes. XML inclut dans le préambule d'un flux de données échangées, la description des éléments de données qui composent ce flux.

## **Interopérabilité – Un exemple simple**

---

Pour illustrer ce qu'est l'interopérabilité, prenons un exemple simple qui correspond aux besoins classiques des établissements publics dans le monde.

Imaginons que le système A gère les permis de conduire ; il a été développé il y a de nombreuses années. Le système B gère les accidents de voiture. Le système B a besoin d'accéder aux données du système A. Plusieurs solutions existent :

1. Dupliquer les fichiers du système A sur le système B.
2. Le système B interroge le système A pour obtenir les informations dont il a besoin.
3. Le système B interroge et met à jour les données du système A (solution plus compliquée).

Ces trois scénarios représentent tous un certain degré d'interopérabilité. Toutefois, seule la troisième solution peut être considérée comme une véritable interopérabilité dans un seul sens.

Comment bâtir cette troisième solution ? Nous pourrions utiliser un traitement par lots dans lequel les fichiers et les enregistrements seraient traités à intervalle de temps régulier ; ou nous pourrions opter pour des échanges en temps réel. Cela nous conduit à définir différentes solutions à des niveaux différents d'interopérabilité :

- Dupliquer les fichiers du système A sur le système B.
- Traitement par lots des données.
- Traitement des données en temps réel (mode interrogation).
- Traitement des données en temps réel (mode mise à jour).
- Traitement des données en temps réel (mode bidirectionnel de mise à jour).

De plus, nous sous-entendons que cette interopérabilité est prise en charge par un ensemble d'outils de communication standards qui permettent d'envoyer et de lire des données de façon transparente d'un système à un autre, indépendamment du système d'exploitation, des protocoles réseau et des langages de programmation des applications.

Aujourd'hui, cela revient à largement utiliser le protocole TCP/IP à la base des infrastructures de télécommunications et des réseaux. Le processus de télécommunication de base étant en place (TCP/IP), il est possible d'y ajouter de nouvelles couches, comme des composants intermédiaires, pour permettre des communications plus simples et plus efficaces entre applications.

## **Interopérabilité et passerelles : Étendre les possibilités**

---

Dans l'exemple précédent, nous avons supposé que les développeurs du système B connaissent bien le système A, du moins suffisamment pour écrire le code nécessaire pour accéder directement aux données du système A. Toutefois, dans la réalité, de nombreux systèmes sont en production, le code source n'est pas toujours disponible et l'équipe de développement d'origine a disparu depuis longtemps.

C'est un scénario habituel dans le monde réel. De plus, écrire des applications qui traitent directement des données n'est pas une bonne pratique en termes de programmation, notamment quand ces données appartiennent à un autre système. La solution qui résout ces deux problèmes à la fois consiste à mettre en place une couche intermédiaire entre les applications afin d'établir le niveau de sécurité requis pour l'interopérabilité. L'application A parle à un système intermédiaire et ce système intermédiaire parle à son tour à l'application B. Cette application intermédiaire agit comme un interprète polyglotte : il connaît tous les langages et tous les dialectes de toutes les applications concernées. Parfois, il sera nécessaire d'écrire des « connecteurs » dépendant des applications pour parler à l'application intermédiaire.

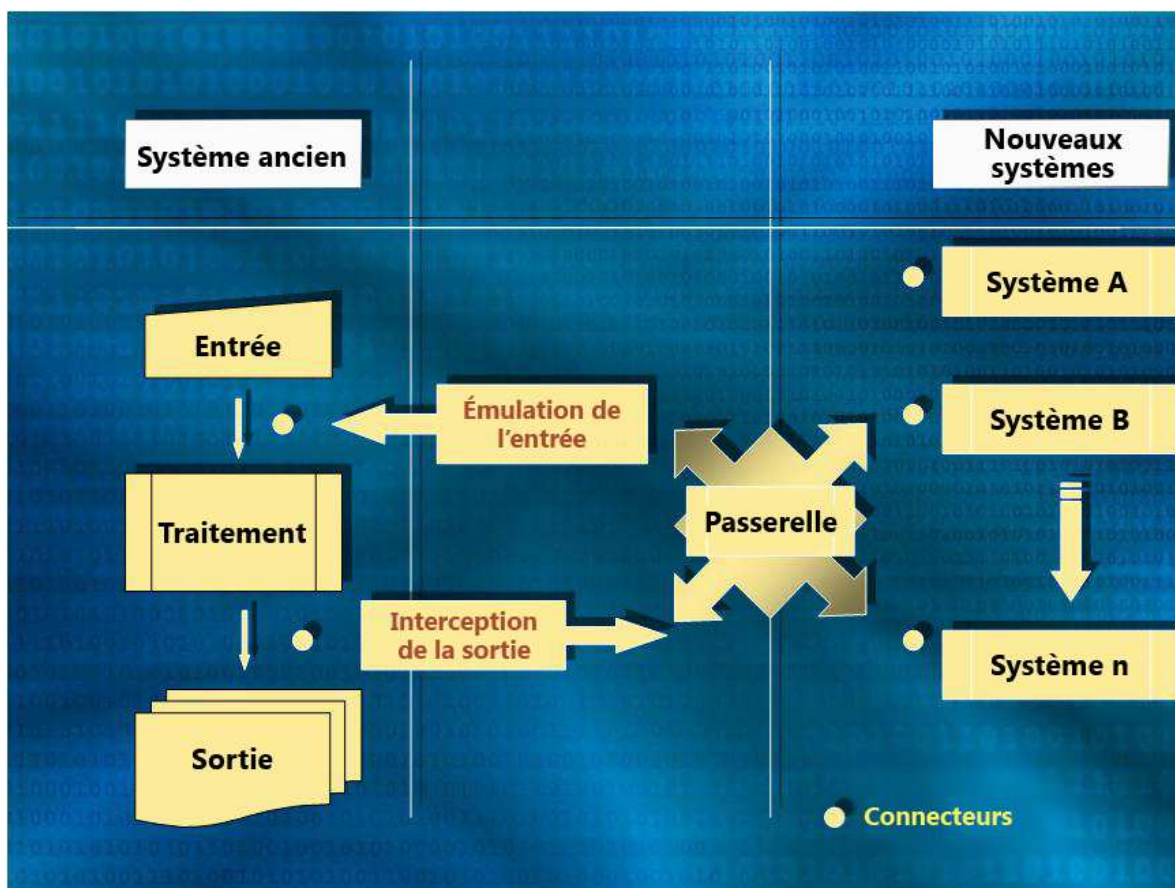
Le grand avantage de ce concept est que nous avons maintenant un seul point où tous les langages et tous les dialectes sont définis. Si un système change sa structure de données internes, il suffira de mettre à jour uniquement son connecteur ou l'application intermédiaire. Techniquement, cette application intermédiaire se nomme « passerelle d'interopérabilité ».

Utiliser une telle passerelle présente encore d'autres avantages. Les systèmes centraux anciens constituent encore aujourd'hui une part importante des solutions existantes. La plupart de ces systèmes n'était pas conçu pour prendre en charge les protocoles et les technologies actuelles. Leurs capacités d'interopérabilité sont restreintes en raison des intentions de leur conception originelle.

Si un système ancien doit échanger des informations avec un autre système, cela peut conduire à réécrire l'application ancienne, à écrire une application en parallèle ou alternative, ou encore à concevoir une solution de pont entre les deux systèmes. Tout se simplifie si nous fournissons un mécanisme capable d'émuler les formats de données des systèmes anciens.

Tous les systèmes informatiques sont basés sur trois fonctions de base : les entrées, le traitement et les sorties. Si, en utilisant des technologies existantes, nous pouvons intercepter les flux de données en entrée et en sortie, nous devrions pouvoir assurer une interopérabilité avec n'importe quel système.

La meilleure façon d'atteindre cet objectif consiste à produire les données nécessaires en entrée dans le format et avec la méthode d'entrée utilisée par le système ancien. Par ailleurs, les données en sortie seront interceptées et pourront être transmises à l'autre système.



Cette méthode peut être généralisée pour assurer l'interopérabilité entre tous les systèmes. La solution de la passerelle est similaire à un traducteur de données qui reçoit les données du système source, les traduit et les transmet à un ou plusieurs systèmes destination. Le système destination traite alors la transaction demandée et produit les résultats désirés.

C'est exactement là où la solution de la passerelle s'applique, car en plus d'être un traducteur de langages et de formats utilisés par les applications, la passerelle peut aussi développer des fonctions d'émulation pour intercepter les données en provenance ou à destination du système ancien.

Une passerelle et son connecteur adéquat peuvent émuler un ancien terminal vidéo et automatiquement entrer des données dans une application ancienne, comme si un utilisateur réel saisisait les données. Dans le même temps, la passerelle peut se comporter comme une imprimante et capturer des données à partir de l'ancienne application afin de les transmettre au système moderne dans le format approprié.

Prenons l'exemple suivant :

Le système X est un mainframe IBM utilisant CICS et le gestionnaire de base de données ADABAS. Sa mission est de gérer les enregistrements et les inscriptions des étudiants d'une université. Une des fonctions transactionnelles mettant en œuvre les terminaux texte 3270 est d'interroger le système en fournissant le numéro d'identification d'un étudiant afin de trouver à quels cours cet étudiant est inscrit.

Dans la même université, la bibliothèque a décidé de développer un nouveau système basé sur Microsoft .NET et SQL Server afin de suivre l'utilisation de la bibliothèque par les étudiants via une application Web. Le nouveau logiciel doit déterminer si le numéro d'identification de l'étudiant est valide ou non afin de pouvoir prêter un livre au demandeur.

La nouvelle application obtiendra le numéro de l'étudiant via un formulaire Web et transmettra la donnée à la passerelle assurant l'interopérabilité. La passerelle convertira les données des standards XML/TCP/IP/UNICODE en standards SNA/LU2/EBCDIC et transmettra le résultat à l'application IBM. Cette dernière comprendra les données reçues comme si elles provenaient d'un terminal 3270, et y répondra. La passerelle interceptera les données renvoyées par l'application IBM et les convertira en sens inverse pour les transmettre à l'application Web.

La passerelle jouant le rôle d'une application intermédiaire qui comprend et traite les deux côtés de la transaction, l'interopérabilité sera assurée entre les deux systèmes même si le système ancien n'était pas prévu pour le Web au départ.

En outre, la passerelle offre des avantages supplémentaires. Il n'est pas rare dans les architectures anciennes d'avoir des données dupliquées dans différentes bases de données. Cela se produit car les différentes applications ont été écrites indépendamment les unes des autres, sans souci d'intégration. Prenons l'exemple d'applications dans le secteur public pour gérer des cartes d'identité, des permis de conduire ou des dossiers fiscaux. Il est bien possible que l'adresse d'une personne soit stockée trois fois, dans chacun de ces systèmes apparemment sans lien entre eux. L'adresse n'est pas partagée entre les systèmes et il n'existe aucun contrôle de cohérence et aucune assurance sur la qualité des informations collectées. Par conséquent, si la personne déménage, elle devra effectuer plusieurs démarches auprès de différents organismes pour indiquer à chaque fois la même information : sa nouvelle adresse. Dans une architecture « interopérable », la passerelle peut gérer d'elle-même la mise à jour des différents systèmes, ce qui réduit les incohérences. Si un individu met à jour ses données personnelles, une architecture intelligente peut automatiquement propager les nouvelles informations dans tous les systèmes, assurant ainsi un meilleur service au public et une utilisation plus efficace des ressources. Toutes les parties impliquées y gagnent.

La passerelle introduit aussi un autre avantage : elle peut constituer un mécanisme d'identification unique de l'utilisateur dans des systèmes hétérogènes. En raison de l'historique du développement des systèmes anciens, il est habituel que chaque système définisse son propre mécanisme d'authentification afin de permettre à un individu d'utiliser une application. Par conséquent, un utilisateur doit fournir plusieurs identifiants pour exploiter toutes les applications dont il a besoin. En revanche, si une passerelle est installée, elle peut stocker tous les identifiants et fournir automatiquement les droits d'accès à n'importe quel système auquel l'utilisateur a le droit d'accéder. L'utilisateur ne communique qu'une fois son identifiant principal et la passerelle en déduira tous les autres. Elle fédère ainsi différents identifiants et permet une « ouverture de session unique ».

## Interopérabilité et convergence

La convergence définissait à l'origine la tendance de fusion entre les traitements des données et les télécommunications. Au départ, ces technologies étaient totalement distinctes. Elles sont aujourd'hui parfaitement réunies. Toutefois, la convergence évolue et s'applique désormais à d'autres domaines. Les processeurs et les ordinateurs ont été intégrés dans de nombreux équipements qui s'intègrent à leur tour dans des réseaux. Des téléphones cellulaires peuvent envoyer et recevoir des courriels et interagir en temps réel avec d'autres systèmes. Les appareils domestiques ont aussi été intégrés. Un réfrigérateur peut passer commande à un magasin sur Internet sans intervention humaine. Une automobile peut proposer une meilleure route à son conducteur en combinant un logiciel, un GPS, un téléphone mobile et des requêtes Web. C'est ça aussi, la convergence.

Un ERP traditionnel (ou PGI), écrit pour gérer des milliers de commandes transmises par des clients, n'était pas conçu au départ pour recevoir des commandes d'un réfrigérateur ou pour envoyer à un téléphone portable un message signalant l'envoi d'une commande. Or, ce genre d'interaction est aujourd'hui nécessaire, et nous avons besoin de solutions simples et génériques pour y répondre.

Par conséquent, lorsque nous parlons interopérabilité, nous ne devons pas oublier la convergence. Aujourd'hui, les systèmes ont besoin d'échanger des informations avec d'autres systèmes mais aussi avec toutes sortes d'appareils et d'équipements.



L'élément clé pour faciliter la convergence réside là aussi dans des solutions logicielles capables de gérer les besoins d'interopérabilité que la convergence exige. Et ces solutions ne peuvent exister que si elles mettent en œuvre des standards et des protocoles reconnus et acceptés.

Les composants de la famille Microsoft Windows (XP, Vista, Embedded et Mobile) et .NET Framework créent l'infrastructure qui permet le développement de solutions convergentes et interopérables.

## SOA et services WEB

---

À mesure que nous avançons dans l'utilisation de la technologie et que l'interopérabilité devient une réalité, la stratégie pour développer un système change totalement. Aujourd'hui, les systèmes interagissent et l'omniprésence des réseaux est due à la très large adoption d'un protocole réseau de télécommunication, TCP/IP.

Un système ne se résume plus à un logiciel énorme et monolithique, composé de millions de lignes de code, s'exécutant sur un gros ordinateur central. Aujourd'hui, il est possible d'avoir de nombreux systèmes totalement intégrés, avec des centaines de sous-systèmes fonctionnant sur des ordinateurs éparpillés partout dans le monde.

Si tel est le cas, pourquoi faudrait-il réécrire un composant d'application qui fonctionne déjà correctement, quelque part sur le réseau ? Prenons un exemple simple. Un programmeur développe une application efficace qui accepte un code postal en entrée et renvoie toutes les localités possibles pour ce code. Cette fonctionnalité devrait pouvoir s'intégrer facilement dans n'importe quel logiciel qui en a besoin, sans devoir être réécrite à chaque fois. De plus, le programme qui réalise effectivement cette fonction devrait pouvoir s'exécuter n'importe où, la fonction étant proposée sous la forme d'un service facturé à l'utilisation.

Un système peut ainsi être construit sur un réseau de nombreux fournisseurs de services, nommés les services Web. L'architecture qui englobe ces concepts se nomme « architecture orientée service » ou SOA. La SOA peut être techniquement définie comme « une approche pour organiser l'informatique de telle sorte que les ressources d'infrastructure, la logique et les données soient utilisées via des messages échangés entre des interfaces réseau ».

Il faut donc établir des interfaces cohérentes et stables, frontales à diverses implémentations, afin d'établir un contexte pour l'échange d'informations entre organisations. SOA requiert l'engagement de toute l'entreprise pour bâtir toutes ses applications autour d'un modèle de composants services.

L'un des avantages les plus importants dans l'adoption d'une stratégie de services Web/SOA réside dans la diminution de la complexité d'un déploiement système et dans la réduction des coûts de maintenance, dès lors que des systèmes peuvent être construits en assemblant des blocs qui communiquent. Les gros logiciels monolithiques se réduisent désormais à des modules de base qui apportent des services spécifiques à tous les autres membres et composants de la solution complète. Certes, cela s'applique aux nouveaux blocs d'applications. Mais cette stratégie fonctionne aussi pour des applications ou des services existants qui mettent en œuvre des systèmes frontaux de services Web.

Toutefois, l'introduction de ces technologies s'accompagne de défis importants. Les services Web sont promis à un grand avenir mais nous devons nous poser des questions :

- Comment assurer des connexions fiables sur des réseaux qui peuvent tomber en panne ?
- Comment sécuriser les connexions ?
- Comment créer des applications qui travaillent sur plusieurs zones de confiance ?
- Faut-il programmer pour fournir des services ou en bénéficier ?
- Réseau entre homologues et/ou réseau d'entreprise ?

Le passage d'un développement traditionnel à une approche SOA impose de nouvelles relations entre les différentes parties qui fournissent les services. Nous pouvons comparer ce changement à la décision d'externaliser des services. Il faudra s'entendre avec le fournisseur pour garantir un niveau de service minimal via un contrat qui couvrira les services et l'infrastructure.

Certes, il est toujours possible de développer les principales applications métier comme d'habitude mais l'architecture globale des applications devra évoluer pour faire face à de grands changements conceptuels. La complexité relative à la sécurité, à la fiabilité et au routage sera déplacée vers une nouvelle couche interface. Le tableau suivant décrit la situation dans les grandes lignes.

## Modifier le concept de l'architecture des applications pour passer à une architecture orientée service (SOA)



## Un standard ouvert n'est pas « open source »

---

Souvent, les termes « standard ouvert » et « open source » sont confondus comme s'il s'agissait de la même chose. Il n'en est rien. Par exemple, selon le document « Road Map for Open ICT Ecosystems » écrit par le Berkman Center for Internet & Society à Harvard Law School, « ...les standards ouverts ne doivent pas être confondus avec les logiciels open source ». (pg. 6) <http://cyber.law.harvard.edu/epolicy> (2005).

Un standard ouvert est une spécification technique (c'est-à-dire un ensemble d'instructions techniques et de conditions requises) qui présente les caractéristiques suivantes :

1. Spécification développée, maintenue, approuvée ou affirmée par consensus dans une organisation de normalisation et de standardisation, à l'écoute du marché et ouverte à tous les participants intéressés et qualifiés.
2. Publiée sans restriction (sous forme électronique ou autre), avec suffisamment de détails pour permettre une compréhension totale de l'objectif et de la portée du standard (par exemple, une entreprise intéressée doit pouvoir mettre en œuvre ce standard sans rencontrer de zones d'ombre).
3. Accessible à tous gratuitement ou moyennant un paiement raisonnable, pour l'adoption et la mise en œuvre par toute partie intéressée.
4. Tous les droits sur les brevets nécessaires pour mettre en œuvre les standards ouverts sont rendus accessibles par ceux qui développent le standard afin que toutes les personnes et entreprises intéressées puissent exploiter ce standard, moyennant un contrat aux termes raisonnables et non discriminatoires (RAND) et le versement éventuel de royalties.

Parfois, le logiciel open source (OSS) est de façon erronée confondu avec l'interopérabilité : l'utilisation de logiciels open source assurerait l'interopérabilité. En fait, dans certains cas, l'inverse peut être vrai. Tous les codes source OSS pouvant être modifiés par n'importe qui, un produit open source qui, à l'origine, respectait certains standards et assurait l'interopérabilité avec d'autres produits, peut être modifié et ne plus respecter les standards et l'interopérabilité. À l'extrême, la liberté de modifier le code encourage la création de nombreuses variantes de l'application de départ, ce qui oblige à multiplier les tests d'interopérabilité.

Microsoft a développé et met en œuvre des centaines de standards propriétaires et de standards ouverts dans ses produits pour améliorer leur interopérabilité avec d'autres produits et services. Pour preuve des progrès de Microsoft dans ce domaine, une étude menée en 2003 par Lawrence Associates/Forbes montre que Windows présente une amélioration de 102% par rapport aux produits concurrents open source dans le respect des standards.

## **Standards et interopérabilité : un grand mariage**

---

Comme nous l'avons mentionné précédemment, depuis le début de l'informatique dans les entreprises dans les années 1960, l'une des premières exigences des clients consistait à faire dialoguer différents matériels entre eux. Chaque fournisseur de matériel et chaque éditeur de logiciel possédait ses propres protocoles de communication et, par exemple, il était très difficile de faire fonctionner l'imprimante du fournisseur A avec des systèmes autres que ceux de A.

La situation est restée la même pendant longtemps, même après l'arrivée du PC. Sous MS-DOS, il était toujours complexe d'ajouter une imprimante à un PC. Chaque application avait besoin de son propre pilote d'impression, même pour un simple texte.

Windows 3.0 a développé le concept WYSIWYG sur le marché des PC et supprimé la nécessité d'avoir des pilotes dans chaque application pour chaque imprimante. Avec Windows 3.0, le pilote devient une fonctionnalité fournie par le fabricant de l'imprimante, Windows offrant un protocole standard pour qu'une application accède à une imprimante. Cette stratégie a simplifié l'écriture des applications car les éditeurs de logiciels et les fabricants de matériels n'avaient plus qu'à exploiter l'interface standard de Windows. Les fabricants de matériel n'avaient plus à prendre en compte toutes les API des applications et les développeurs n'avaient plus à programmer les dialogues avec toutes les interfaces propriétaires des fabricants. Tout était pris en charge par le système d'exploitation Windows.

Toutefois, il était toujours difficile de connecter physiquement les matériels aux PC. De nombreuses spécificités techniques et des demandes d'interruption devaient être prises en compte et il fallait installer manuellement les pilotes. Ce n'était pas une tâche simple et seuls des utilisateurs connaissant bien le matériel y parvenaient.

Avec Windows 98, Microsoft a développé le concept Plug-and-Play. Ce standard, immédiatement adopté par les fabricants de matériels et de logiciels, a considérablement simplifié l'installation de matériels sur le PC. Il suffit de connecter le nouveau matériel et quelques dizaines de secondes plus tard, l'installation est terminée. Le principe repose sur le code unique qui accompagne le périphérique et l'identifie auprès du système d'exploitation. Ce dernier peut alors lancer les programmes nécessaires pour automatiquement installer et configurer l'équipement.

Tous les problèmes n'étaient pas encore résolus pour autant. Certains équipements nécessitant des échanges à haut débit avec le processeur, il fallait ouvrir le PC pour y installer des cartes internes. La solution apparut sous la forme d'un nouveau standard nommé USB (Universal Serial Bus) qui, par un simple connecteur externe, a permis d'établir des communications à haut débit sans ouvrir le PC.

Aujourd'hui, à quelques exceptions près, presque tous les équipements peuvent se connecter en quelques secondes à un PC et commencer immédiatement à échanger des informations. Imprimantes, scanners, appareils photo, caméscopes, équipements audio, capteurs, téléphones mobiles, presque tout peut être facilement relié à un PC grâce au concept Plug-and-Play et à quelques standards du marché comme USB, Bluetooth et IEEE 1394 (« Firewire »).

Il s'agit là du plus bel exemple d'interopérabilité réalisé dans l'ensemble de tous les secteurs d'activité, et pourtant, c'est le moins utilisé pour expliquer pourquoi les standards sont indispensables pour assurer l'interopérabilité entre différents systèmes, composants et systèmes. Lorsque les choses deviennent simples, nous avons tendance à oublier comme les choses étaient difficiles avant que le problème ne soit résolu.

## Conclusion

---

Le modèle actuel replaçant le serveur au centre de l'environnement informatique ne reflète pas l'incroyable puissance d'une architecture informatique distribuée. En général, les systèmes anciens n'étaient pas conçus pour fonctionner ensemble, ce qui explique pourquoi l'interopérabilité peut parfois être complexe, coûteuse et gourmande en ressources.

Il serait possible :

- D'établir un nouveau modèle de programmation basé sur une architecture orientée service et utilisant des standards largement adoptés par le marché.
- De proposer à un prix très abordable un ensemble d'outils qui implémenterait ce nouveau modèle de programmation, en prenant en compte les compétences existantes et en améliorant dans le même temps l'efficacité.
- D'obtenir les avantages d'un réseau performant, grâce à une interopérabilité assurée entre les systèmes anciens et les nouvelles applications grâce à l'utilisation de services Web.

Nous avons vu qu'il existe de nombreux moyens d'assurer l'interopérabilité (par exemple, lors de la conception des logiciels, en donnant accès à la propriété intellectuelle via des licences, en établissant des collaborations entre entreprises, et en adoptant des standards du marché). Ce livre blanc a montré comment les protocoles et les standards représentent les éléments essentiels pour assurer l'interopérabilité entre différents systèmes. Le dispositif de passerelle est un élément important de ce processus. Ensemble, tous ces éléments permettent de masquer la complexité de chaque système et de son langage interne pour nous permettre de nous concentrer sur le cœur du problème.

Bien que d'autres langages existent, XML est rapidement devenu la langue universelle pour les échanges de données et l'interopérabilité entre systèmes. Il constitue aujourd'hui un standard important. De nombreux pays ont adopté XML comme standard pour assurer l'interopérabilité entre agences publiques, mais peu ont réellement compris tout le potentiel et les avantages qu'ils peuvent tirer de l'adoption de XML et du concept de passerelle.

L'architecture orientée service (SOA) et les services Web permettent des déploiements à faible coût, simplifient la maintenance des applications et créent une base solide pour accroître l'interopérabilité entre des systèmes hétérogènes et géographiquement dispersés.

Les gouvernements peuvent largement profiter de l'adoption d'une stratégie d'interopérabilité au sein d'une administration électronique, en s'appuyant sur XML, SOA et les services Web. Ces technologies peuvent contribuer à développer la transparence et l'efficacité des services publics, à la fois en interne et vis-à-vis des contribuables, dans des domaines comme la fiscalité, les soins, la sécurité, la justice et l'éducation. En mettant en place une infrastructure d'administration électronique efficace, un gouvernement apporte de meilleurs services à ses citoyens et à ses administrations tout en réduisant les coûts.

Bien sûr, il ne faut pas s'attendre à ce que l'implémentation de ces technologies soit simple. Toutefois, le déploiement réussi d'une architecture orientée service et de services Web est moins un problème de technologie qu'un problème de gestion et d'organisation. Une bonne préparation est nécessaire pour mettre en place ce nouveau degré d'interopérabilité entre systèmes et services. De nombreuses barrières, tant culturelles que politiques, doivent être dépassées dans chaque organisation, en plus de la résolution des problèmes purement techniques.

SOA repose sur des contrats de service et des échanges de messages. La spécification de ces contrats entre services doit être claire et le contenu des messages doit être défini avec précision pour réussir le déploiement et l'exploitation d'une telle architecture. Les données doivent circuler au sein de toute l'organisation mais nous constatons souvent que les données ont des significations différentes d'un département à l'autre au sein de la même organisation. C'est la raison de l'échec de la plupart des projets informatiques complexes. Il faut définir un répertoire identifiant toutes les données et les métadonnées ; les modèles de gouvernance et ce répertoire doivent être acceptés par toutes les entités concernées. Ce n'est pas un problème purement informatique mais un élément clé du succès de tout projet informatique.

Le futur de ces nouvelles technologies est prometteur. Comme c'était le cas avant les standards Plug-and-Play et USB, nous constatons actuellement un comportement fermé dans le domaine du logiciel. Dans un futur proche, nous disposerons des outils et des technologies qui permettront de développer des applications

en se concentrant sur le cœur du service à réaliser. Tous les autres éléments, comme la sécurité, les services de bases de données, les accès réseau, ne feront plus partie de l'application : ils seront devenus des services qui utiliseront des standards bien définis et largement acceptés, encadrés par des contrats de niveaux de service ; notre application se contentera de « louer » ces services en fonction de ses besoins. Comme pour USB, nous développerons une application et nous la mettrons simplement sur le réseau ; les autres applications de l'environnement la reconnaîtront et pourront l'utiliser de façon transparente.

Microsoft s'est engagé dans la voie de ces nouvelles technologies. XML et les services Web sont au cœur de tous les produits Microsoft, et de nombreux autres standards informatiques du marché sont également pris en considération. Par exemple, pour jouer le rôle de passerelle, Microsoft propose BizTalk® Server, une solution générique pour assurer l'interopérabilité entre systèmes. Cette combinaison de produits et la prise en charge des protocoles largement utilisés sur le marché placent Microsoft en bonne position pour assurer l'interopérabilité entre logiciels et systèmes.

## Glossaire

---

API	Application Programming Interface – Nom générique utilisé pour décrire le protocole devant être respecté pour communiquer avec une application.
ASCII	American Standard Code for Information Interchange. Codage numérique sur 7 bits des caractères anglo-saxons et de codes de contrôle.
CICS	Customer Information Control System – Système IBM pour le traitement des transactions en ligne dans les sites centraux.
SGBD	Système de gestion de base de données
EBCDIC	Extended Binary Coded Decimal Interchange Code. Codage des caractères sur les systèmes IBM.
ERP	Enterprise Resource Planning, ou PGI (progiciel de gestion intégré).
HTTP	Hypertext Transfer Protocol. Protocole fondamental du Web.
LU.2	Logical Unit Type 2 – Une des spécifications SNA pour communiquer avec les terminaux 3270.
MVS	Multiple Virtual Space – Système d'exploitation IBM.
ODBC	Open Database Connection.
SDLC	Synchronous Data Link Channel – Un des composants SNA qui permet le transport d'informations sur le réseau.
SNA	System Network Architecture – Standard IBM pour l'architecture réseau.
SOA	Architecture orientée service.
SOAP	Simple Object Access Protocol – Protocole définissant la syntaxe pour accéder à des services.
TCP/IP	Transmission Control Protocol/Internet Protocol.
UDDI	Universal Description, Discovery and Integration.
UNICODE	Universal Code.
USB	Universal Serial Bus.
Web	Fait référence au World Wide Web.
WSDL	Web Services Description Language – Fournit le contrat pour des services Web, définissant les entrées attendues et les sorties produites.
XML	eXtensible Markup Language – Pour l'interopérabilité des données entre systèmes, indépendamment des éditeurs de logiciels.

## **Bibliographie**

---

1. SNA, IBM's Networking Solution ; James Martin, 1987.
2. DNS and BIND ; Paul Albitz & Cricket Liu, 1997.
3. ABABAS Reference Manual ; Software AG.
4. E-government – O governo eletrônico no Brasil ; Florência Ferrer & Paula Santos, 2004.
5. América Latina Puntogob ; Rodrigo Araya D. etc... 2004.
6. « Government Interoperability – Enabling the Delivery of E-Services » ; Jerry Fishenden, Oliver Bell, Alan Grose. Livre blanc Microsoft, avril 2005.
7. « Road Map for Open ICT Ecosystems » ; Berkman Center for Internet & Society at Harvard Law School, septembre 2005
8. BSA Statement on Technology Standards, février 2005.
9. Interoperability: Definition, How to Achieve, Choice as Best Policy, Microsoft's Commitment; Microsoft Corporation, décembre 2005.
10. Windows or Linux – Evaluate the total cost before you decide ; Lawrence Associates ([http://www.lawrence-associates.com/Downloads/Public/Windows%20or%20Linux%20TCO\\_wp%20\(0903\).pdf](http://www.lawrence-associates.com/Downloads/Public/Windows%20or%20Linux%20TCO_wp%20(0903).pdf))
11. 2005 World Development Indicators Database ; World Bank, 16 avril 2005.
12. The Global Information Technology Report, 2004-2005 ; Soumitra, Dutta and Augusto Lopez-Claros, World Economic Forum Reports – INSEAD.
13. Three Unexpected Results in Open-Source Software Engineering; Stephen R. Schach, Vanderbilt University, Nashville, TN, United States.
14. WITSA – Digital Planet: The Global Information Economy (<http://www.witsa.org/news/99mar.htm>).
15. Gartner – 2005 Web Services Platform Magic Quadrant Report.