Communauté, modularité et efficacité dans la livraison de logiciels
Livre blanc
Juillet 2009

Rational. software

IBM

L'avenir de la livraison de logiciels.

Danny Sabbah, Directeur Général PhD,
Rational Software

Sommaire

Introduction

« *Chaque génération se moque de l'ancienne mode, mais suit religieusement la nouvelle* »

—Henry David Thoreau

De nos jours, les professionnels de la conception et du déploiement de logiciels ne peuvent pas ignorer les tendances actuelles : architecture orientée services (SOA), logiciels basés sur une communauté (logiciels en open source [OSS]) et développement réparti à l'échelle mondiale (GDD).[1] Selon un analyste, « 9 entreprises sur 10 adoptent ou ont adopté une architecture orientée services … elles en auront déjà réalisé la planification, la conception et la programmation »[3] Les logiciels OSS devraient faire partie des applications de mission critique de plus des trois quarts des grandes entreprises internationales d'ici la fin de la décennie, et de nombreux sondages indiquent qu'une majorité d'entre nous utilise des logiciels OSS en production dès *maintenant*.[4]

Bien que les concepts SOA, OSS et GDD semblent être des phénomènes sans précédent, souvenons-nous qu'ils ne viennent pas de nulle part. Avant de nous engager dans l'adoption d'une nouvelle architecture, d'un nouveau modèle logiciel ou d'une nouvelle méthode de création et de maintenance de logiciels, il est important de bien prendre en compte ce que SOA, OSS et GDD représentent et d'où ils viennent. Cela nous permettra de mieux nous préparer à leur éventuelle adoption de masse.

Points clés

Le processus est le principal facteur d'avancement.

Le concept d'architecture SOA n'est pas totalement nouveau. La mise au point de services homogènes est l'idée fondatrice du développement de la technologie Enterprise JavaBeans Java™ , et avant cela de CORBA. L'architecture SOA est ouverte, basée sur les normes, orientée communauté, gouvernable, modulaire et simple (certains de ses concepts existent depuis longtemps). Pour OSS, c'est la même chose. En fait, dans le contexte du développement logiciel, il n'existe aucune invention sous-jacente derrière le concept OSS.

On utilise depuis 25 ans des logiciels en Open Source(ils proviennent en grande partie du monde universitaire), mais depuis cinq à sept ans, ils se sont considérablement renforcés, principalement en raison d'une tendance très importante : l'émergence de logiciels basés sur une communauté. Les fonctions les plus visibles d'OSS (innovation, adoption large, coût réduit) sont revisitées. Une véritable valeur ajoutée est apportée à la livraison de logiciels via le processus suivant, en trois étapes :

1. Les modules logiciels satisfaisants favorisent la participation de la communauté à créer dans le domaine OSS.
2. L'innovation et la banalisation des composants sont de mise dans les environnements de développement ouverts.
3. La normalisation conduit à l'adoption par l'industrie.

Par conséquent, le produit peut bien être OSS, c'est le processus qui constitue le principal facteur d'avancement (c'est cela, les logiciels basés sur une communauté).

Ce qui compte vraiment, ce n'est pas SOA, OSS, GDD ou autres TLA, c'est plutôt leurs préceptes et processus sous-jacents : communautés d'intérêt, systèmes modulaires et efficacité. En s'appuyant majoritairement sur les capacités induites par les tendances (et non simplement sur les mots à la mode), nous pouvons sans plus attendre commencer à incorporer ces avantages dans nos logiciels (si ce n'est pas déjà le cas).

Points clés

IBM œuvre pour l'incorporation de communautés plus larges de spécialistes.

L'incorporation des exigences et des méthodologies actuelles à notre propre livraison de logiciels (en diffusant les meilleures pratiques testées et les innovations récentes pour répondre à des besoins précis) est le thème de ce livre blanc. IBM Rational® a déjà tiré parti de ces leçons via la création et la maintenance des produits et solutions existants ; IBM concentre actuellement ses efforts sur l'extension de ces principes dans les versions futures des produits Rational, afin de favoriser la participation de communautés de spécialistes plus vastes.

Les autres parties de ce document se composent de quatre sections : brève description des problèmes passés en matière de livraison de logiciels, ayant contribué à l'environnement actuel ; identification des principales tendances qui guident notre activité à l'heure actuelle ; présentation de la stratégie du groupe Rational en matière du futur de livraison de logiciels ; synthèse succincte.

Développement de logiciels Legacy

De nos jours, la réalité du développement de logiciels et de systèmes est la suivante : il peut être plus rapide de construire une nouvelle usine de fabrication que de déployer un nouveau système ERP (Enterprise Resource Planning). Il est plus rapide, en apparence, d'intégrer un fournisseur de pièces détachées au sein d'une chaîne logistique physique qu'au sein d'une chaîne logistique informatique.

Par exemple, en mai 1986, Toyota a fait figure de pionnier avec sa nouvelle usine d'assemblage de Georgetown, Kentucky. Moins de deux ans plus tard (23 mois, pour être précis), la première voiture sortait de la ligne d'assemblage. En revanche, la durée d'installation moyenne d'un système ERP SAP est de trois ans (33,6 mois, très exactement).[5] La création d'une structure de routage des artefacts numériques, des idées et des processus autour d'un système informatique prend plus de temps que la construction d'une usine d'assemblage, l'approvisionnement de pièces de voiture en tous points du globe et la mise sur le marché de la première voiture.

Points clés

Le secteur du développement logiciel en est à son adolescence.

Le secteur du développement de logiciels a maintenant plus de 50 ans d'existence ; toutefois, l'écart existant entre l'implémentation physique et l'implémentation logicielle reste vaste. Le problème réside dans le fait que ce secteur, ainsi que les processus logiciels, ne sont pas encore totalement formés : ils ne sont pas matures. En effet, ce secteur en est seulement à son adolescence, et connaît une forte croissance. L'une des causes principales de cette volatilité est la base d'exécution de nos logiciels : la base de notre infrastructure informatique connaît une évolution encore plus rapide. En outre, l'augmentation de la vitesse des unités centrales a conduit à une certaine inefficacité dans notre mode de développement et de déploiement de logiciels.

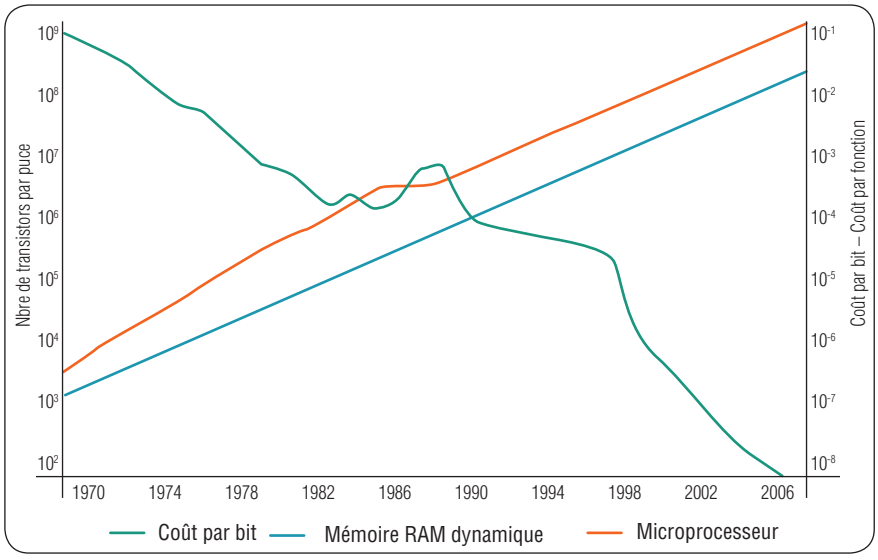
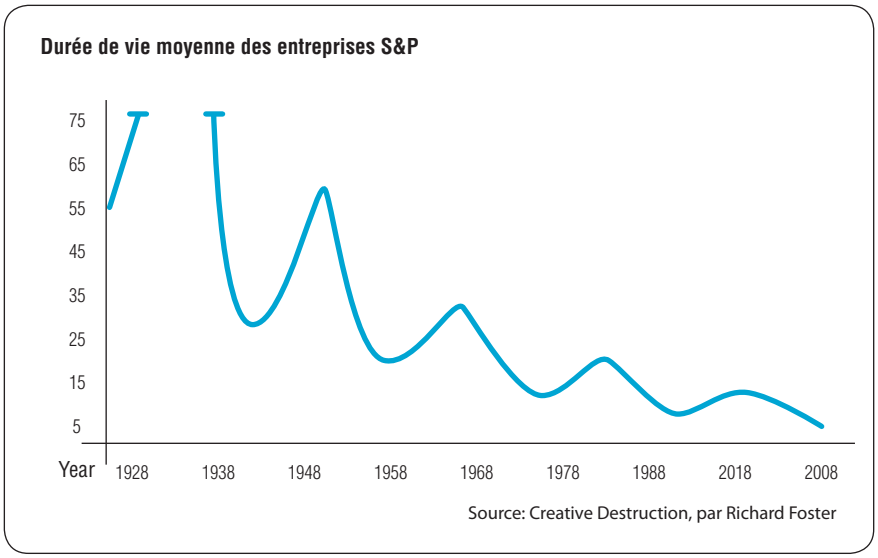


Figure 1. La puissance de traitement et de mémoire augmente, les prix diminuent.

La Figure 1 illustre l'évolution de la vitesse des processeurs, (sur la base des énoncés de la loi Moore,[6]), ainsi que la chute correspondante des prix des ressources informatiques. La rapide évolution des composants matériels et logiciels permet de bénéficier de nouveaux champs d'applications de la technologie ; nous pouvons donc effectuer des fabrications et des déploiements de logiciels qui n'auraient pas été *possibles* il y a seulement quelques années. Désormais, en raison des faibles coûts de la mémoire et des capacités de traitement, une myriade de nouveaux cadres de références logicielles (frameworks) peut tirer profit de ces avancées.

Points clés

Les applications existantes qui ne peuvent pas s'intégrer à ces nouvelles infrastructures sont désavantagées ; elles agissent elles-mêmes comme des facteurs d'inhibition du changement.



La durée de vie moyenne des entreprises S&P a considérablement diminué depuis 1930.

A l'heure actuelle, le marché se présente de façon totalement différente d'il y a 5 ou 10 ans.

Au fur et à mesure de l'augmentation de la vitesse des technologies informatiques et de communication, la durée de vie moyenne d'une entreprise S&P 500 évolue. En 1930, la durée de vie moyenne était de 75 ans ; aujourd'hui, elle est tombée à 15 ans. L'évolution rapide du marché mondial et les principes de la capitalisation portent en leur sein des leçons importantes à prendre en compte. A l'heure actuelle, le marché se présente de façon totalement différente d'il y a 5 ou 10 ans, et il est tentant de dire que toutes les règles ont changé. Or, ce n'est pas le cas.

Points clés

Souvenons-nous de l'idée de nouvelle économie, qui date de la fin des années 1990, et des ratios P/E évoqués dans les années 2000, qui ont fait apparaître une nouvelle réalité dans la sphère de la valorisation des entreprises. Puis en 2001, l'inexactitude de tout cela a été massivement prouvée. Les entreprises se créent et se dispersent à une vitesse accrue, mais (du moins pour l'instant), il semblerait que les anciennes règles régissent encore la valorisation du marché.

Cette même leçon peut s'appliquer pour bien comprendre comment aborder la ruée actuelle vers les avancées technologiques. Nous traversons, et c'est logique, une tornade de technologie informatique ; il est donc facile de perdre de vue notre expérience dans le domaine du commerce, de la construction, du développement de logiciels et de l'intégration de systèmes. En prenant un peu de recul, nous pouvons facilement voir ce qui se trouve dans le sillage de cette tornade.

Les méthodologies d'ingénierie logicielle, les paradigmes de gestion de projets, les langages de programmation et de génération de scripts, et enfin les cadres de références d'entreprise sont tous concernés par ce flux.

L'architecture logicielle n'a pas encore atteint le même niveau de maturité que l'architecture traditionnelle – du batiment, par exemple ; de la même façon, la technologie de chaîne logistique logicielle n'a pas non plus atteint le même niveau de capacités et de robustesse que la chaîne logistique physique. Les méthodologies d'ingénierie logicielle, les paradigmes de gestion de projets, les langages de programmation et de scriptings, et enfin les cadres de références d'entreprise sont tous concernés par ce flux. Toutefois, comme nous l'a appris la Bourse au cours de ces 10 dernières années, les principes qui régissent les processus conduisant à la vraie valeur ajoutée tendent à rester les mêmes.

Faire ressortir la vraie valeur des méthodologies et des processus existants, comprendre où nous devons concentrer nos énergies et identifier si nous sommes sur la bonne voie sont des clés qui nous révèlent les éléments à conserver et ceux à éliminer, parmi toutes les possibilités qui s'offrent à nous, de nos jours. IBM Rational a identifié un certain nombre de tendances faisant apparaître une vision stratégique qui mérite que l'on s'y penche.

Points clés

Accès réseau abordable et réseaux sociaux ont eu un effet majeur.

Tendances actuelles importantes

Si l'on fait un bref retour en arrière, au tout début de ce 21ème siècle, il est certain que l'un des phénomènes ayant eu le plus d'impact sur la communauté de développement logiciel et sur le monde entier est l'avènement d'un accès réseau abordable et des réseaux sociaux rendus possibles grâce à Internet et aux technologies sans fil. L'arrivée des téléphones portables dans les pays en voie de développement, par exemple, a déjà eu un effet majeur sur l'économie de ces pays. Et cette avancée s'est faite exclusivement via des transactions verbales. Imaginez ce qui se passera lorsque ces sociétés étendront leur commerce à Internet et commenceront à développer la propriété intellectuelle, en se libérant des frontières physiques traditionnelles de temps, d'espace et de capitaux.

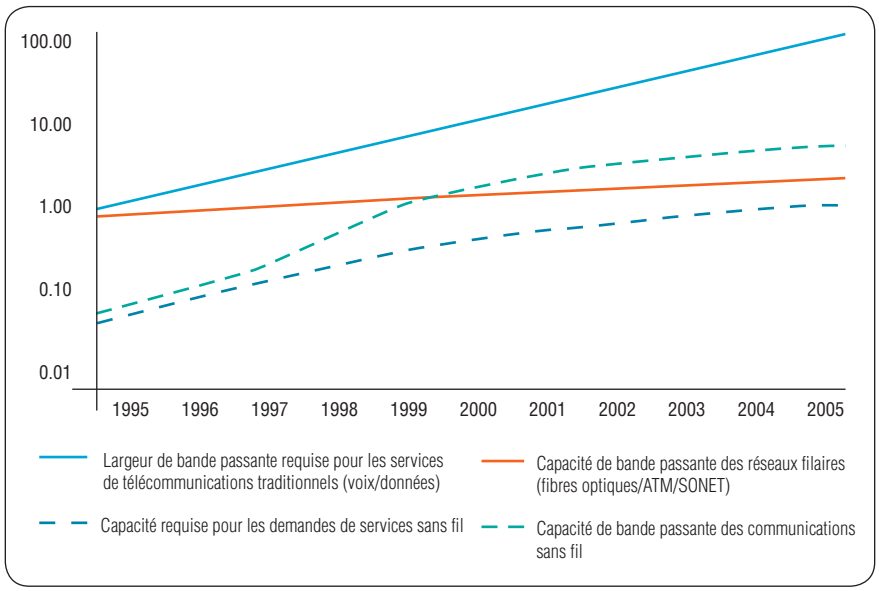


Figure 3. La largeur de bande passante continue d'augmenter.

Points clés

La bande passante continue d'augmenter : nous assistons actuellement au début de ce phénomène. L'un des résultats de l'augmentation de la bande passante sur les différents supports est l'importance des données transmises (ou du contenu de ces données), par rapport au mécanisme lui-même. Le secteur des télécommunications a déjà pu constater une convergence des flux de données et d'informations ; le débat n'est plus centré sur la prévalence de la télévision, d'Internet ou de la téléphonie. Le débat tourne aujourd'hui autour de la vitesse, de la fiabilité et de la disponibilité du transfert de données ; il revient à l'utilisateur final de décider de l'emplacement, du format et de l'unité de destination du contenu transmis.

En d'autres termes, l'accès aux données est en train de changer nos vies. La Loi Metcalfe7 indique que la valeur d'un réseau est proportionnelle au carré du nombre d'utilisateurs du système $\frac{n(n-1)}{2}$: cela signifie que plus le nombre de personnes connectées est élevé, plus la valeur générée par les connexions est grande. Et les réseaux sociaux en apportent déjà la preuve.

Les nouvelles communautés ou les sous-groupes se rassemblent autour d'idées intéressantes et novatrices.

En effet, les communautés ont des agendas spécifiques, et possèdent des objectifs communs transparents : pour Linux®, l'intérêt réside dans les bibliothèques ; pour Eclipse, l'intérêt réside dans les projets. Actuellement, les développeurs de logiciels ont de nombreuses opportunités de participation à des communautés existantes, voire de création de leur propre communauté. En l'absence de structure formelle de gouvernance, ces communautés se multiplient et sont souvent régies par le concept de méritocratie. Les nouvelles communautés ou les sous-groupes se rassemblent autour d'idées intéressantes et novatrices. Toute personne ayant suivi ces discussions pourra témoigner de la fermeté avec laquelle sont traités les auteurs de code de mauvaise qualité.

Points clés

Les communautés technologiques sont puissantes : elles incluent la communauté Rational, les utilisateurs Microsoft® , les administrateurs système, entre autres. En fait, ces communautés constituent, de par leur taille, la plus grande force de définition de normes de facto. Citons la communauté de développeurs en langage C, les développeurs Java, les utilisateurs PHP, les développeurs de sites Web, qui ont façonné non seulement leur propre domaine, mais aussi la façon dont nous créons les logiciels. Ce qui fait la différence aujourd'hui, c'est la vitesse à laquelle ces communautés se forment. Alors qu'il a fallu 10 ans au langage C pour atteindre le niveau de popularité que nous lui connaissons, 5 ans ont suffit au langage Java pour faire de même. Et en 2 ans d'existence seulement, la technologie relativement nouvelle Subversion devient l'héritier de la technologie CVS (Concurrent Version System), vieille de 20 ans.

Ce qui fait la différence aujourd'hui, c'est la vitesse à laquelle ces communautés se forment.

Mais la vitesse n'est que l'une des dimensions caractérisant les communautés d'aujourd'hui. Le point de rupture intervient, comme toujours, au moment où un nombre important de Dirigeants décide que cette technologie possède tout ce dont ils ont besoin.[8] L'un des aspects les plus intéressants du paysage dans lequel nous évoluons réside dans le fait qu'un tout petit groupe de personnes peut constituer le point de rupture (un changement peut même être motivé par les besoins d'une seule personne).

Examinons maintenant un certain nombre de tendances qui sont apparues avec l'avènement de la bande passante et de la connectivité : architecture orientée services (SOA), logiciels basés sur une communauté et environnement régissant leur succès. Au cœur de ces tendances, nous trouvons un certain nombre de principes (et notamment le processus de distillation, réduisant une technologie complexe à son essence, ce qui revient à appliquer le principe logique Occam's razor (principe de parcimonie) aux problèmes logiciels rencontrés. Par exemple, il est important de se souvenir que Linus Torvalds n'a pas inventé de système d'exploitation, mais qu'il a simplifié les modules UNIX® existants. L'architecture SOA, couvrant de nombreuses technologies

Points clés

complexes, en constitue un concept très simple : les services homogènes atomiques. Cette tendance communautaire mène de la complexité à la clarté, et guide toutes les tendances décrites dans ce document. IBM Rational a fait ressortir que les tendances logicielles porteuses de succès s'articulent autour de trois caractéristiques fondamentales : communauté, modularité et efficacité. Nous détaillerons chacune de ces caractéristiques.

Développement de logiciels basé sur une communauté

- Evolution de la programmation[9]

La programmation au sens large a évolué.

Si vous souhaitez entamer un débat philosophique sur le thème « Combien d'anges peuvent danser sur la tête d'une épingle ? », vous trouverez une multitude d'exemples de ce type au sein des blogs. Rédigez l'énoncé suivant : "What constitutes *real* open source software?" (Quels éléments constituent les véritables logiciels en Open Source?) Vous découvrirez vite qu'il existe très peu d'infrastructures pouvant s'enorgueillir de la dénomination OSS "pur". Même la technologie Linux (pourtant considérée comme la technologie Open Source la plus aboutie au monde) est divisée en plus de 350 distributions (selon les derniers chiffres en notre possession10), affichant différents niveaux de source ouverte. Quant à Ruby, cette technologie est principalement sous le contrôle d'une seule personne au Japon, et tout le monde recherche des financements. Dans ce domaine, un long chemin a été parcouru depuis FSF (Free Software Foundation). Il importe surtout de ne pas se laisser enfermer dans des arguments de type religieux, mais de juger chaque infrastructure logicielle et chaque produit en fonction de ses caractéristiques. Commençons par examiner la définition de base d'OSS.

Ouverture

La meilleure définition des logiciels en Open Source est, comme son nom l'indique, l'ouverture. La technologie OSS vous permet de consulter le code source. Les avantages d'un accès au code d'origine sont implicites, mais non garantis, comme nous allons le voir.

Points clés

Facile à modifier, facile à corriger

Cette modification n'est pas garantie par toutes les infrastructures OSS, car il existe plus de trois douzaines de schémas de licences OSS (chacun d'entre eux étant légèrement différent) ; pensez donc à tout ce que vous êtes obligé de faire une fois que vous avez commencé à manipuler le code ! Les modifications peuvent également avoir un impact négatif sur les contrats de support. Et la liberté a ses limites : en effet, la possibilité de redéfinir les logiciels est une bénédiction pour les développeurs, mais elle représente un casse-tête pour les responsables informatiques, qui doivent gérer le nouveau logiciel après le départ du développeur, le cas échéant.

Facilement Disponible

Pendant des décennies, les fournisseurs de logiciels ont tenu leur code source secret, car ils pensaient (et pensent encore, pour la majorité d'entre eux) qu'ils perdraient leur avantage concurrentiel si tout le monde avait accès à leurs travaux internes. Ce secret a eu un effet négatif sur l'adoption de logiciels, nécessitant une force de vente et de nombreux documents marketing pour retirer le voile sur cet aspect. La source, dès qu'elle est rendue publique, peut être rapidement disponible pour tous, réduisant à zéro le prix de vente, et autorisant chacun à évaluer les capacités réelles du logiciel. Les logiciels en Open Source ne sont jamais des logiciels shelfware (non utilisés).

Le véritable coût d'OSS commence après l'installation.

Gratuit

L'expérience pratique nous indique que les logiciels en Open Source sont gratuits, tout comme souvent les chiots. Le véritable coût d'OSS commence après l'installation, avec l'arrivée des coûts de développement, de déploiement et de maintenance. Cela n'est pas nouveau en soi : c'est la même chose pour tous les logiciels. Les logiciels OSS modifiés induiront des coûts plus élevés liés aux aspects ci-dessus, car les solutions isolées sont à caractère unique. Nous avons pu voir que le modèle dominant des entreprises OSS qui marchent bien consistait à gagner de l'argent via l'offre de support technique, de services et de formation, et cela a du sens. En d'autres termes, le coût réside davantage dans les soins et l'alimentation, comme pour notre exemple sur les chiots. Comme avec un chiot, un mauvais choix OSS peut entraîner un coût élevé par la suite.

Points clés

Modularité

Dans OSS, la modularité n'est pas implicite, mais elle signe généralement la marque de tout bon logiciel. Cela inclut des lignes de démarcation claires, des interfaces API raisonnables et un cadre de référence extensible capable d'inclure des composants enfichables. Les meilleurs logiciels OSS sont modulaires, comme le sont les meilleurs logiciels du commerce.

Basés sur une communauté

Il s'agit là du cœur des logiciels OSS performants, ainsi que du cœur de la conception et de l'exécution des logiciels de qualité en général. Les communautés et les normes ouvertes coexistent, dans la mesure où une communauté dominante sera généralement à l'origine de la norme de facto correspondante. Le problème ici consiste à garantir la meilleure norme possible, et dans ce contexte, la clé réside dans la gouvernance exercée à différents niveaux. Le maintien d'un système de suivi des corrections sain, l'encouragement d'un débat vivant, le développement de normes utiles, tous ces éléments sont le résultat d'une gouvernance raisonnable appliquée à chaque niveau.

Tout cela est le résultat d'une gouvernance raisonnable, appliquée à chaque niveau.

En résumé, OSS peut contenir de nombreux attributs caractérisant les logiciels de qualité, mais ce n'est pas une obligation : modularité, disponibilité, coût raisonnable (prix et maintenance), normes ouvertes, robustesse, performances et communauté d'utilisateurs correctement gérée. Même si ces qualités ne sont pas réservées aux logiciels en Open Source, elles représentent des caractéristiques idéales, quel que soit le logiciel. Sur ce point, nous pouvons nous rallier à l'esprit du débat sur les développeurs OSS décrit plus haut. Nous recherchons des logiciels possédant ces qualités positives, mais le secteur doit encore se mettre d'accord sur l'obtention de ces qualités.

Points clés

Architecture orientée services

L'architecture SOA partage de nombreux thèmes communs avec les thèmes développés ci-avant autour des logiciels OSS. L'architecture SOA représente essentiellement une architecture de systèmes d'informations permettant de créer des applications via la combinaison de couplage faible et de services interopérables. Ces services présentent une interopérabilité prenant comme base une définition formelle (ou contrat) indépendante de la plate-forme et du langage de programmation.

Les services SOA doivent être bien définis (encapsulés) , faciles à retrouver et posséder un couplage faible, afin d'être suffisamment autonomes, abstraits (réutilisables) et capables de s'intégrer à des services plus complexes, possédant eux-mêmes les attributs ci-dessus.

Il est facile de tomber dans le piège qui consiste à dire que l'architecture SOA correspond à une technologie spécifique.

Les normes SOA sont encore en cours de définition ; par conséquent, il est facile de tomber dans le piège qui consiste à dire que l'architecture SOA correspond à une technologie ou à une solution spécifique. Il y a quelques années, beaucoup de personnes pensaient *SOA = Services Web*, car les services Web répondent aux exigences principales des services SOA. Cette association *SOA ≠ Services Web* est évidente aujourd'hui, mais la même erreur se rencontre parfois dans le postulat *SOA = ESB* (Enterprise Service Bus). Avec les autres technologies, l'équation reste fausse. En fait, les services Web et ESB sont des outils très utiles au sein d'une architecture orientée services correctement constituée. Des services Web mal agencés et des modèles mal conçus créés via ESB en l'absence d'une architecture SOA ne sont que des allers simples vers la catastrophe.

Points clés

L'architecture SOA se compose de 1 % de services et de 99 % de gouvernance.

L'erreur, bien sûr, serait de croire qu'une seule technologie ou combinaison de normes permettrait d'intégrer l'architecture SOA à votre entreprise. La vérité n'est pourtant pas bien loin. La maxime de Thomas Edison stipulant qu'un génie se compose de 1 % d'inspiration et de 99 % de transpiration est proche de la réalité. Nous pouvons l'adapter et affirmer que l'architecture SOA se compose de 1 % de services et de 99 % de gouvernance.

Une architecture SOA performante, c'est beaucoup de travail, et cela nécessite la concertation des différents départements d'une entreprise, afin d'œuvrer pour le bien de tous. Cette unité est déjà difficile à atteindre dans les petites équipes. Que dire alors de cet objectif ardu dans les entreprises contenant de nombreuses équipes qui doivent collaborer pour la mise au point d'une architecture SOA ?

C'est difficile, mais pas impossible. Une architecture SOA a davantage de chances de succès lorsqu'elle s'appuie sur des normes ouvertes, sur une modularité et sur une communauté bien gérée. Comme pour OSS, l'attribut principal conduisant au succès de tous les autres est la gouvernance. En effet, la gouvernance contrôle la mise en conformité des normes, les contrats de composants modulaires et enfin la capacité de la communauté SOA à travailler ensemble de façon pertinente.

Points clés

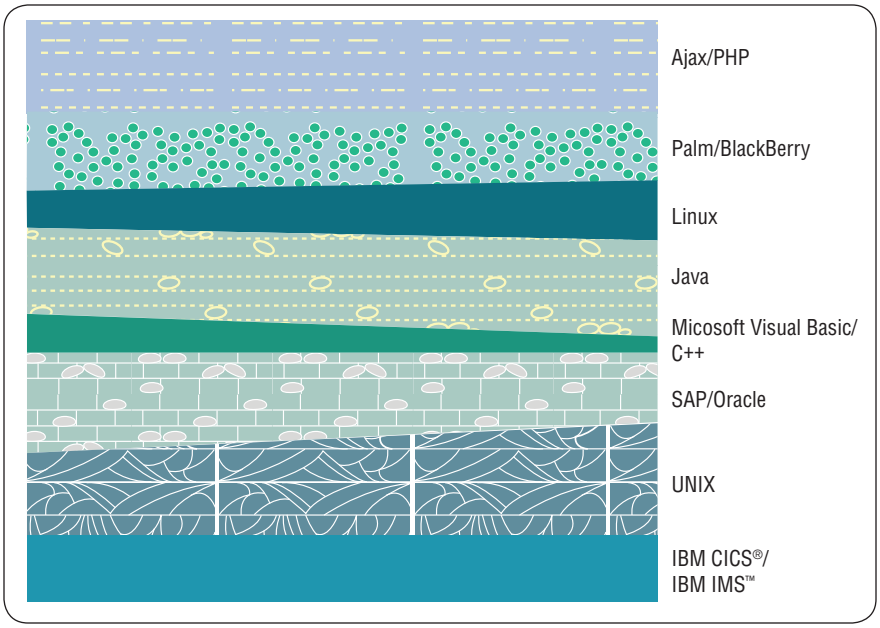


Figure 4. Couches de données, de services et de processus

Il n'est qu'à voir tout ce que le secteur financier et bancaire réalise déjà avec les données SOA.

Dégager les services, les données et les processus ensevelis sous des décennies de couches logicielles nécessite un effort concerté pour récupérer et réaffecter les artefacts qui font la valeur de votre entreprise. Il n'est qu'à voir tout ce que le secteur financier et bancaire réalise déjà avec les données SOA. Comptes, investissements, cartes de crédit, prêts hypothécaires et emprunts peuvent désormais être gérés via une interface utilisateur classique. Ces services sont maintenant interconnectés avec les plans de paiement, les retraits et dépôts automatiques, le renversement de dividendes, les analyses financières, les calculs d'impôt, entre autres. Ces services représentent de nombreux artefacts transmis en toute sécurité aux quatre coins du monde, avec un taux d'échec proche de zéro. Et ce n'est que ce qui est utilisé actuellement dans le secteur bancaire. Il reste encore beaucoup à venir.

Points clés

Comme nous allons le voir, IBM Rational fait son possible pour faciliter votre processus de développement SOA. IBM comprend bien l'importance du développement par des communautés, du partage des informations et d'une bonne gouvernance tout au long du cycle de vie des applications.

Gouvernance et efficacité

Pour beaucoup, la notion de gouvernance fait immédiatement penser à la conformité : celle de Sarbanes-Oxley et de l'HIPAA (Health Insurance Portability and Accountability Act). Elle fait également penser à la non-conformité et à son lot de pénalités pour infraction à la réglementation. L'étendue de la gouvernance, telle qu'elle s'applique à la livraison de logiciels, va au-delà de la mise en œuvre de règles légales. La gouvernance représente davantage qu'une infrastructure de surveillance, qu'un ensemble de normes qui, lorsqu'elles sont mises en œuvre, peuvent favoriser le travail d'une équipe plutôt que l'entraver. Par exemple, le framework de gouvernance de projet présente dans Eclipse a été mis à profit par l'une des communautés en Open Source la plus créative et réactive.[11]

La gouvernance représente davantage qu'une infrastructure de surveillance, qu'un ensemble de normes qui, lorsqu'elles sont mises en œuvre, peuvent favoriser le travail d'une équipe plutôt que l'entraver.

La mise en conformité devient donc le résultat d'une bonne gouvernance, et non le point central de la livraison de logiciels. C, Java, Internet, XML, Linux, logiciels en open source , architecture SOA – : rien de tout cela ne peut exister sans gouvernance. Chacune de ces technologies, de ces architectures et de ces infrastructures a vu le jour précisément en raison des normes en place, qui fixent les limites des comportements, des discussions, des implémentations. Ainsi, un nombre toujours croissant d'utilisateurs peut compter sur la stabilité des technologies et les utiliser efficacement. Le langage C n'aurait pas pu devenir le langage principal sur PC sans cette conformité à la gouvernance ANSI (American National Standards Institute) et ISO (International Organization for Standardization) ; le même principe s'applique à Internet par rapport au consortium W3C (World Wide Web Consortium) et à ICANN (Internet Corporation for Assigned Names and Numbers), ou encore aux solutions Linux par rapport au langage C et à Internet.

Points clés

Les concepts OSS et SOA s'appuient sur ce schéma, et sont tout aussi liés aux exigences de gouvernance que leurs prédécesseurs. Sans gouvernance, tout est trop aléatoire. Le pilotage d'un véhicule à l'aide d'un régulateur de vitesse permet de maintenir la vitesse à la limitation fixée, ce qui n'est pas la même chose que le contrôleur de vitesse, dispositif qui fixe uniquement une limite supérieure. Le régulateur de vitesse illustre bien l'idée de gouvernance efficace, car il garantit un fonctionnement compris dans les limites légales, ce qui maximise l'efficacité tout en minimisant les durées de trajet et les activités triviales. Poursuivons l'analogie en reprenant l'exemple d'un véhicule utilisant la méthode des feux de signalisation chronométrés, ce qui nécessite une gouvernance informatisée. Cela illustre la façon dont nous pouvons utiliser la gouvernance à notre avantage, pour atteindre le plus rapidement et le plus efficacement possible les objectifs que nous nous sommes fixés. Comparons maintenant ce scénario avec une situation que nous redoutons tous : sur la route, lorsque nous voyons une voiture de police postée derrière un arbre. La différence ne devrait pas être plus importante que cela.

Grâce à l'utilisation de la gouvernance à notre avantage, nous nous conformons à une règle, à une infrastructure ou à un produit qui minimise l'effort requis, qui optimise le contrôle implicite, afin de mener à bien un projet. La gouvernance constitue une part essentielle de la livraison de logiciels de qualité, de l'intégration OSS et de la mise en œuvre SOA ; en fait, c'est un aspect très important de chaque facette de la création de logiciels, de son implémentation et de sa maintenance. Il revient à chaque personne ou entreprise d'utiliser à bon escient ces éléments de contrôle.

La gouvernance relie dans l'ombre les différentes communautés.

La gouvernance est le fil transparent qui relie les communautés entre elles. L'existence de chaque communauté repose sur un thème fondateur : des objectifs en commun, des croyances communes, une propriété commune, par exemple. Il existe toujours un contrat implicite qui les unit. Il se crée un grand nombre de communautés, mais peu d'entre elles connaissent une réussite durable.

Points clés

Les principaux ingrédients d'une communauté solide sont la gouvernance et l'efficacité.

Pour qu'une communauté existe assez longtemps pour exercer une réelle influence sur le marché, elle doit être assez solide pour porter la vitalité intellectuelle de ses membres. Les principaux ingrédients d'une communauté solide sont la gouvernance et l'efficacité (soulignons que l'efficacité des individus via la modularité de choix constitue un facteur clé de succès de la livraison de logiciels).

> « *On peut commander le modèle T dans la couleur de son choix… tant que ce choix est le noir !* »
>
> —Henry Ford

Le futur de la livraison de logiciels

Né au début du 20ème siècle, le modèle T de Ford était positionné au bon prix, disponible en quantités de masse et aussi performant que l'annonçaient les publicités. C'était un produit adapté au marché, lancé au bon moment ; toutefois, il n'était absolument pas configurable (disponible en une seule taille, une seule couleur, sans options possibles).

Revenons maintenant à aujourd'hui et examinons l'évolution du secteur automobile : vous pouvez communiquer vos exigences directement à l'usine, et personnaliser en ligne votre nouvelle voiture. Les options sont nombreuses : GPS, radio satellite, lecteur de CD ou MP3, climatisation, détecteurs de pluie, régulation de la consommation de carburant, commandes hybrides, systèmes de contrôle des émissions, systèmes de guidage, systèmes de stationnement automatique, visualisation de l'espace, ainsi que de nombreuses autres options qui justifient le recours à la puissance de l'informatique. J'oubliais : vous pouvez aussi choisir la couleur, bien sûr... L'industrie automobile a intégré des logiciels au sein du châssis et tout au long de la chaîne de fabrication, ce qui lui permet de livrer des produits à la demande.

Points clés

Nous voulons configurer notre technologie en fonction de nos besoins.

Regardez aussi le succès que connaissent les détaillants tels qu'Amazon.com, Netflix et bien d'autres, qui stockent sur d'énormes sites des milliers de produits adaptés à des goûts bien spécifiques. Nous vivons à l'ère de la personnalisation ; nous voulons configurer notre technologie en fonction de nos besoins, et il n'y a pas de raison de faire autrement quand nous disposons de l'intégration système et de la livraison de logiciels sur lesquels s'appuyer. Nous savons que la personnalisation a un prix, et nous acceptons de le payer.

Chaînes logistiques de propriété intellectuelle

Bienvenue dans l'ère de la Longue traîne. La théorie de la Longue Traîne, rendue populaire par le livre de Chris Anderson (vendu sur des sites Web qui attestent chaque jour de la véracité de ses concepts), consiste à dire que l'avenir de l'activité commerciale consistera à vendre moins d'articles généralistes et plus d'articles personnalisés, adaptés à de petits groupes, voire à des personnes individuelles.[12] Pour les fournisseurs de logiciels, cela suppose de développer moins de gros logiciels et plus de composants personnalisables. En bref, cela supposera le développement de cadres de références génériques incluant un grand nombre de plug-in personnalisés.

L'argument de la Longue traîne se focalise plutôt du coté du fournisseur de la chaîne logistique. La technologie permet d'opérer un grand changement dans le modèle de coûts de la chaîne logistique (des boutiques des détaillants aux centres de distribution), ce qui se traduit par un stock plus élevé composé d'éléments disparates. Cela permet au fournisseur de s'approprier la 'longue traîne' de la courbe de la demande en offrant un choix plus vaste.

Points clés

Si l'on applique ce phénomène à la livraison des logiciels, les ramifications de la demande ont des répercussions sur l'offre des fournisseurs de logiciels. D'une certaine façon, l'architecture SOA est à la fois l'expression d'un désir de choisir et un mécanisme remédiant au manque de souplesse des couches sédimentaires des architectures logicielles. Nous parlons ici de l'articulation des informations, des données et des artefacts lors de l'élaboration de chaînes logistiques de propriété intellectuelle.

Il existe un facteur qui renforce la complexité du développement de logiciels : la grande majorité d'entre nous ne tient pas suffisamment compte de la longue traîne des modèles d'utilisation, pour les différentes tâches de développement. Il arrive trop souvent que l'on force certains rôles dans le domaine du développement et des tests via la conformité aux différentes capacités et fonctions des outils que nous utilisons. Les fournisseurs d'outils logiciels déplorent le fait que les testeurs concentrent tous leurs efforts sur le test des performances, et non sur la facette métier de l'application. Mais dans ce cas, les fournisseurs de logiciels portent également une part de responsabilité, car ils mettent sur le marché des outils comportant des fonctions qui encouragent le cloisonnement fonctionnel.

La meilleure façon de créer les produits vient de l'écoute de la communauté.

La meilleure façon d'y remédier consiste à travailler en partant de la demande, afin de bien comprendre et de mettre en place un choix et une souplesse requises pour résoudre le problème. Pour cela, l'écoute de la communauté est primordiale.

Points clés

La technologie Eclipse est intégrée à plus de 200 produits IBM.

Ouverture de la communauté à la participation

L'objectif d'IBM, en mettant Eclipse à disposition de la communauté Open Source en 2001, était de rapprocher les développeurs des middlewares Java, leur ouvrant un monde dans lequel l'environnement de développement d'un client inclut une combinaison hétérogène d'outils. Cela a pu s'effectuer grâce à la création d'un cadre de référence client lourd robuste et à l'accessibilité de tous les services via l'utilisation de plug-in (développés en tant que projets Eclipse, produits tiers ou produits de développeurs indépendants, mais sur une plateforme commune, comprenant un écosystème d'outils logiciels).

La plateforme Eclipse est rapidement devenue la plateforme de développement Java la plus populaire auprès des entreprises du monde entier : elle est actuellement utilisée par 65 à 75 % des développeurs Java.[13] Cela atteste de la popularité d'Eclipse auprès des fournisseurs comme des utilisateurs. La technologie Eclipse est présente dans plus de 200 produits IBM, et sa normalisation au sein d'une plateforme unique de développement et de déploiement a contribué à augmenter son interopérabilité produit.

Les objectifs d'Eclipse en matière de transparence, de capacités de prévision et d'un feedback permanent ont conduit à un niveau sans précédent de solidité des projets (l'un des ingrédients les plus importants dans le domaine de la création de logiciels de qualité). Une communauté saine, un fort esprit d'équipe, des fabrications robustes, des points d'étapes clairs, des versions béta prometteuses et des tests continus : tous ces ingrédients contribuent à la santé florissante du projet Eclipse.

Points clés

Au cours de ces dernières années, l'implication de Rational dans le projet Eclipse a prouvé que la solidité d'un projet était finalement plus importante encore que la qualité des logiciels, et qu'elle était complémentaire. L'état d'un projet est en constante évolution ; par conséquent, le maintien de la solidité du projet devient une responsabilité d'équipe. Et une communauté de développement saine (caractérisée par l'ouverture d'esprit, la responsabilité et l'inclusion) est mieux à même de réagir au stress normal induit par le développement de logiciels en constante évolution : changements non anticipés, nouvelles exigences et cible en perpétuelle mutation.

GLa gouvernance est la clé de la création d'une synergie qui permet de rassembler tous ces facteurs.

Rational thinking

IBM Rational voit l'avenir du développement logiciel comme un milieu impliquant des communautés ouvertes, dynamiques, travaillant dans la plus grande liberté possible tant au niveau des configurations que des personnalisations de solutions modulaires, destinées à des entreprises agiles et réactives. La gouvernance (capacité à exploiter pleinement l'énergie de groupes disparates et de technologies hétérogènes) sera la clé ouvrant la porte de la synergie nécessaire au rassemblement de tous ces facteurs. Aucune norme ou technologie isolée ne pourra dicter la vitesse à laquelle les entreprises de développement feront progresser leurs initiatives métier. Le pilote de ce mouvement ne sera pas l'architecture SOA ou OSS, mais la gouvernance efficace de plusieurs facteurs conduisant à des communautés solides et favorisant la réactivité.

C'est la capacité à obtenir des résultats positifs à un audit gouvernemental sans fournir d'efforts particuliers, car la conformité a été mise en œuvre au sein d'un cadre de référence. C'est le passage fondamental d'un état d'esprit basé sur les différents rôles à une réflexion centrée sur l'équipe. La solution consiste à réaffecter l'équipe, afin qu'elle soit plus impliquée dans toutes les phases qui composent les projets.

Points clés

L'avenir de la livraison de logiciels s'apparente au projet IBM Rational Jazz.

L'avenir de la livraison de logiciels s'apparente au projet IBM Rational Jazz, une technologie qui permet d'intégrer les différentes tâches des équipes tout au long du cycle de livraison des logiciels et des systèmes. Les entreprises qui utilisent l'infrastructure Jazz pourront évaluer l'impact de la modification d'une exigence pour déterminer les conséquences au niveau des fabrications ; cela permettra d'identifier les évolutions à concrétiser, ainsi que les personnes qui en seront chargées. Cette intégration du cycle comprendra un processus s'appuyant sur les outils, car ceux-ci pourront désormais comprendre le processus de développement que l'équipe a décidé d'utiliser. Et cette intégration du cycle permettra, via l'automatisation, de s'assurer que les différents membres de l'équipe peuvent suivre le processus et en rester maîtres.

Pour exploiter pleinement une plateforme de gouvernance du cycle orientée équipe, les entreprises doivent également fournir des composants souples et modulaires, prenant comme base les normes ouvertes utilisées sur la plateforme. Les uns ne peuvent pas exister sans les autres. La décomposition et le découpage en capacités et fonctions existantes sont des opérations parallèles et d'importance équivalente.

C'est un effort de la communauté entière, qui ne peut être réalisé par une seule personne (comme avec Eclipse, en somme). C'est pourquoi, IBM Rational a fait évoluer les logiciels orientés communauté vers le niveau suivant, en contribuant aux projets Linux, Eclipse, Geronimo et autres projets OSS, ainsi qu'aux développements de produits menés par Rational. La capacité d'assistance au processus d'IBM Rational a été conçue pour s'adapter aux différentes échelles de complexités (des pratiques agiles aux approches structurées). Et elle s'adapte également aux différents environnements de développement (des équipes très restreintes aux grandes entreprises réparties). Les entreprises auront la possibilité d'associer des outils et des technologies de livraison de logiciels pour aboutir à une solution optimale, à toutes les étapes de la transformation de leurs environnements de développements .

Points clés

La réponse ? L'agilité.

Conclusion

L'époque actuelle est intéressante, en constante évolution, et nos logiciels suivent cette évolution afin de s'adapter aux besoins accrus du marché. La dure réalité, c'est que nous devons nous adapter ou disparaître, et la menace d'obsolescence ne se compte plus en années, mais en mois. Cela ne signifie pas que tout a changé, bien au contraire. Les créateurs de logiciels font face aux mêmes réalités : la modélisation logicielle et le recueil des exigences métier restent des activités essentielles, et la qualité élevé des logiciels est toujours la meilleure solution possible. Ce qui a changé, bien sûr, c'est la vitesse, l'étendue et la portée de l'activité.

Les solutions ne sont pas totalement nouvelles, mais elles intègrent de nouvelles idées. Il devient essentiel de personnaliser votre livraison de logiciels pour répondre aux exigences des clients, de plus en plus segmentées ; toutefois, vous courez à l'échec si vous concentrez tous vos efforts sur la mise sur le marché de la énième fonction. Vous devez au contraire faire preuve de la plus grande agilité : vous devez articuler les données, les ressources et les artefacts pertinents au sein de vos couches de propriété intellectuelle actuelles, car votre solution contiendra à coup sûr un mélange de types, marques et couches logicielles différents.

Points clés

La livraison de logiciels est régie par trois principes clés : communauté, modularité et efficacité.

Pour que vos données travaillent pour vous, vous devrez refondre vos artefacts existants et les réaffecter au sein de votre entreprise. Cela signifie que vous devez rester réactif et adaptable, afin que vos clients puissent individualiser leur intéraction avec votre business. Dans ce domaine, le succès dépend de votre aptitude à suivre de très près les trois principes clés de l'avenir de la livraison de logiciels : communauté, modularité et efficacité.

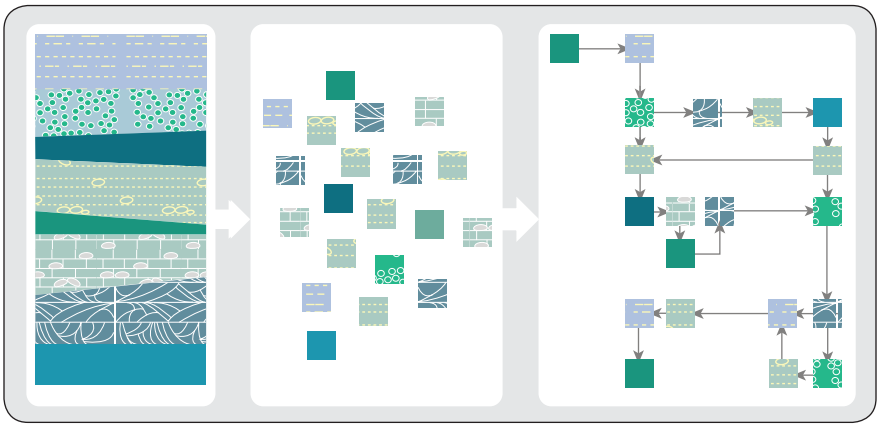


Figure 5. Mise en évidence de couches de données, de services et de processus, et utilisation de ceux-ci

A l'avenir, la livraison de logiciels se rapprochera de la notion de chaîne logistique : organisation horizontale faiblement couplée mais régie efficacement pour répondre à vos exigences SOA et GDD, en faisant apparaître des données, des processus et des services dont vous avez besoin pour obtenir un retour sur investissement positif. L'avenir sera moins fait d'applications individuelles, de systèmes d'exploitation ou de logiciels à choisir mais plutôt de livraison de services, de modularité des systèmes et d'implication de votre communauté bien définie et flexible dans les prises de décisions stratégiques. IBM Rational est en marche pour faire de cette vision une réalité. Communauté, modularité

Points clés

et efficacité guident les produits Rational actuellement développés, ainsi que l'infrastructure Jazz et toutes les offres futures de produits Rational.

L'avenir de la livraison de logiciels s'annonce palpitant. Et vous pouvez être certains que IBM Rational continuera à jouer un rôle prépondérant dans ce domaine.

IBM Rational est en marche

Pour en savoir plus

Pour en savoir plus à propos de des produits IBM Rational de livraison de logiciels, contactez votre représentant commercial ou visitez :

ibm.com/software/rational

Notes

1 Friedman ,Thomas L., *The World is Flat: A Brief History of the Twenty-first Century;* 2005; Farrar, Straus et Giroux.

2 Lo Giudice, Diego, *Service Oriented Architecture: The Foundation for Digital Business;* Forrester Research; 2006.

3 *Enterprise Service Bus and SOA Middleware;* Aberdeen Group; Juillet 2006.

4 *Open Source in Global Software: Market Impact, Disruption, and Business Models;* IDC; Juillet 2006.

5 Nucleus Research various ROI reports; 2003 (http://nucleusResearch.com).

6 http://en.wikipedia.org/wiki/Moore's_Law.

7 http://en.wikipedia.org/wiki/Metcalfe's_law.

8 Gladwell, Malcolm, *The Tipping Point;* 2002; Little, Brown & Co.

9 http://en.wikipedia.org/wiki/Programming_in_the_large.

10 http://distrowatch.com.

11 Erich Gamma and John Wiegand, *The Eclipse Way;* 2006.

12 Pour consulter d'autres exemples du postulat de longue traîne, visitez le site : http://www.longtail.com/the_long_tail.

13 Cette figure inclut l'utilisation d'autres produits intégrant des composants Eclipse au sein de leurs environnements IDE commerciaux (IBM Rational Application Developer, par exemple).

Compagnie IBM France
Tour Descartes - La Défense 5
2, avenue Gambetta
92066 Paris La Défense Cedex

Imprimé en France
07-09

RAW10983-FRFR-00