Java, langage et architecture

Fiche technologique

OCTOBRE 2000

LE CIGREF

Le Cigref, Club informatique des grandes entreprises françaises, existe depuis 1970. Sa finalité est la promotion de l'usage des systèmes d'information comme facteur de création de valeurs pour l'entreprise. Il constitue un lieu privilégié de rencontre et d'échange d'informations entre les responsables des grandes entreprises françaises ou européennes utilisatrices d'importants systèmes d'information. Ce partage d'expériences vise à faire émerger les meilleures pratiques. Chaque année, le Cigref réalise des études sur des sujets d'intérêt commun.

Rapports publiés par le Cigref en 2000 :

Gérer les connaissances
Défis, enjeux et conduite de projet

Impacts et usages de la messagerie électronique

La sécurité à l'heure d'internet

Le phénomène Linux en entreprise (à paraître)
Fiche technologique

Mobilité et GSM (à paraître)
Fiche technologique

Nomenclature 2000 (édition de septembre 2000)
Les emplois-métiers du système d'information

Observatoire 2000 des Télécoms

XML, vers un format universel ?
Fiche technologique

Ces rapports peuvent être obtenus en se connectant sur le site web du Cigref : www.cigref.fr

PARTICIPANTS

Un groupe de réflexion animé par Guy Lapassat, directeur informatique de la Générale des Eaux, a été constitué au Cigref, avec la participation active des personnes et entreprises suivantes :

Thierry Allembach	Lyonnaise des Eaux
Georges Arhodakis	L'Oréal
Jean –pierre Asun	AtoFina
Yves Barthelemy	Michelin
Jean-Claude Baux	Alcatel
Henri Bénoliel	SMABTP
Jacques Bisiaux	Crédit foncier de France
Alain Bonjean	Grepac (Agirc)
Paul Bourgmayer	Alcatel
Christian Cadé	Cnav-TS
Elisabeth Canat	CNCA
Anne-France Chambon	Banque de France
Martine Chicault	Radio-France
Bertrand de Greef	AP-HP
Ky Do Ngoc	Framatome
Marc Dukat	Cnav-TS
Brigitte Genovese	Alstom
Alain Gérard	CNCE
Philippe Gillot	Caisse des Dépôts et Consignations
Claude Gnemmi	MGEN
Aline Grasset	MMA
Jacky Grinenwald	Crédit lyonnais
Jean-Claude Hurteau	Axa
Pascal Laurent	Mairie de Paris
Thomas Lemele	MMA
Olivier Lenormand	CNRS
Patrick Lepage	L'Oréal
Jean Philippe Madelaine	Cogema
Victor Emmanuel Maduro	Intermarché
Gérard Margueritte	Framatome
Danièle Mermet	Crédit foncier de France
Jean-Michel Michl	PSA
Olivier Mimaud	SNCF
Jacques Nussli	MGEN
Jean-Marc Pailloux	SNCF
Alain Paoli	Intermarché
Marc Persuy	PSA
Jean-Marie Pilot	Cnav-TS
Laurent Poulalion	Agirc
Jeannine Pugin	Framatome
Ulrich-André Renauldon	Axa
Gilbert Rochard	AP-HP
Marc Rocher	Mairie de Paris
Marie-Françoise Rotenberg	Société générale
Vincent Russo	Alstom
Yves Soussan	Mairie de Paris
Jèrome Topezenki	Natexis Banques populaires
Gilles Tréhin	ParisBourse^{SBF} SA
Jean-Jacques Vaultier	Azur GMF
Fabrice Viger	Natexis Banques populaires
Christian Vouillon	Framatome
Hiep Vu Thanh	AP-HP
Philippe Zanini	Mairie de Paris

L'étude a été rédigée par Guy Lapassat (Générale des Eaux), Stéphane Rouhier (Cigref) et Frédéric Lau (Cigref).

SOMMAIRE

1. RÉSUMÉ

Java est un langage de programmation inspiré de C++, géré par des « machines virtuelles », environnements d'exploitation qui s'interfacent avec les systèmes d'exploitation des serveurs.

La quasi-normalisation de Java et l'architecture décrite ci-dessus permettent d'utiliser Java sur de nombreux ordinateurs et de rendre les programmes écrits en Java largement indépendants des systèmes d'exploitation sur lesquels ils tournent.

Java n'est pas un langage de description de pages statiques comme HTML ou dynamique comme XML : c'est du code applicatif qui s'exécute sur un serveur.

2. ENJEUX

Actuellement, Java est un standard géré par Sun ainsi qu'une communauté de développeurs. Cette communauté est ouverte et attentive aux tentatives de dévoyer ou circonscrire le langage.

Cette ouverture est la garantie de la pérennité d'un standard qui permet de faire des applications indépendantes des plates-formes : le langage Java permet de s'affranchir pour une bonne part des stratégies commerciales des fournisseurs.

Cette relative indépendance permet à une entreprise d'envisager plus facilement de développer et d'utiliser des composants logiciels métiers portables d'un environnement à un autre, qu'il s'agisse de progiciels ou de logiciels spécifiques.

3. HISTORIQUE

Sun est à l'origine du langage Java. Le projet date de 1992 sous l'appellation « *Green Project* ». À cette époque, deux anciens responsables de Sun cherchent à développer un système d'exploitation (OS) baptisé « Oak », destiné à l'informatique domestique (télévision, assistant personnel…).

Le projet échoue en 1994 et l'OS domestique est abandonné, mais Bill Joy (Sun) retravaille avec ses équipes le langage qu'il rebaptise « Java ». C'est en utilisant le langage de programmation pour internet que Java va réellement décoller à partir de 1994 (machine virtuelle Java, compilateur, API, sécurité…).

À partir de 1996, des éditeurs comme Netscape commencent à intégrer des composants Java dans leur offre (navigateur), suivis en 1997 par Microsoft, IBM…

4. DESCRIPTIF

Dans les années 80-90, les architectures utilisaient des solutions largement propriétaires (systèmes d'exploitation du serveur, poste de travail, *middleware* — protocole de communication —...).

Ces architectures posent des problèmes avec internet, dans la mesure où les postes de travail des utilisateurs sont extérieurs à l'entreprise, c'est-à-dire qu'ils peuvent utiliser des systèmes d'exploitation divers et des versions autres que celles utilisées par l'entreprise qui offre des services internet.

Les possibilités d'avoir des programmes indépendants des composants techniques du poste de travail présente donc un intérêt considérable.

La solution était d'avoir des applications auxquelles on puisse accéder *via* une interface générique comme par exemple un navigateur, mais composées de code qui s'exécuterait soit en local sur le poste de travail, soit sur le serveur. Ce code devrait respecter, bien entendu, un standard non propriétaire et reconnu.

De là est né le langage Java qui est un langage de programmation objet inspiré de C++, reposant sur des classes et sur la notion d'héritage. Il permet d'utiliser les plates-formes informatiques, les interfaces homme-machine (IHM), le réseau et les bases de données de façon unique et homogène : il favorise donc la portabilité des développements et s'affranchit des notions propriétaires évoquées plus haut.

De l'avis des utilisateurs, c'est un langage peu encombrant, sûr et portable.

D'autre part, deux évolutions ont renforcé l'attrait de Java :

- l'évolution d'internet de services d'information statiques, pour lesquels HTML était un outil satisfaisant, à des services d'information dynamiques (pages intégrant des informations extraites de bases de données) et à des services transactionnels ;
- l'utilisation des architectures internet sur d'autres terminaux que les PC (téléphone, TV, NC...).

Dans le premier cas, la gamme des possibilités de Java permet de satisfaire des besoins fonctionnels.

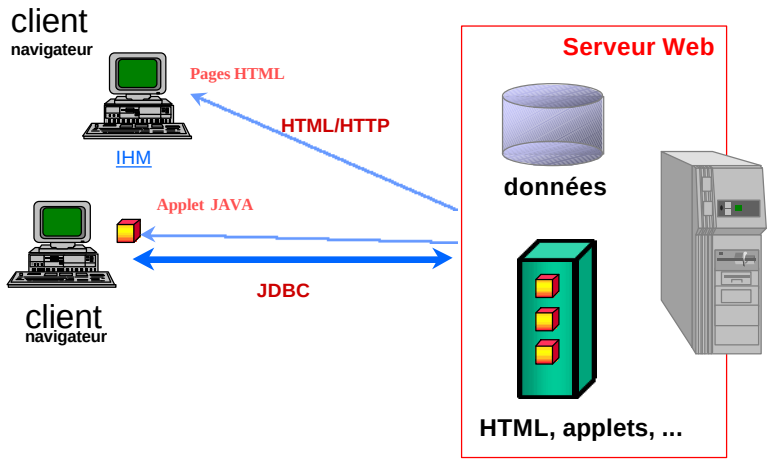
Dans le second cas, la portabilité de Java répond aux besoins techniques.

Une application Java n'est pas une suite de balises permettant de décrire un traitement simple comme HTML ou XML, mais bien un code informatique exécuté sur la machine cible. Pour cela, toute application Java doit s'exécuter dans un environnement d'exécution pour accéder à des composants métiers qui interrogent les ressources système *via* des interfaces :

- l'environnement d'exécution est *appelé « Java Virtual Machine »* ou JVM ;
- les interfaces d'accès aux ressources sont regroupés dans le *« Java Development Kit »* ou JDK ;
- les composants métiers sont appelés aussi des *« beans »*.

Progressivement, une architecture autour de Java s'est mise en place autour de ces trois briques de base. Elles permettent de mettre en œuvre des applications exécutées sur le poste de travail et sont alors nommées *« applet »* ou sur un serveur et nommées *« servlet »*.

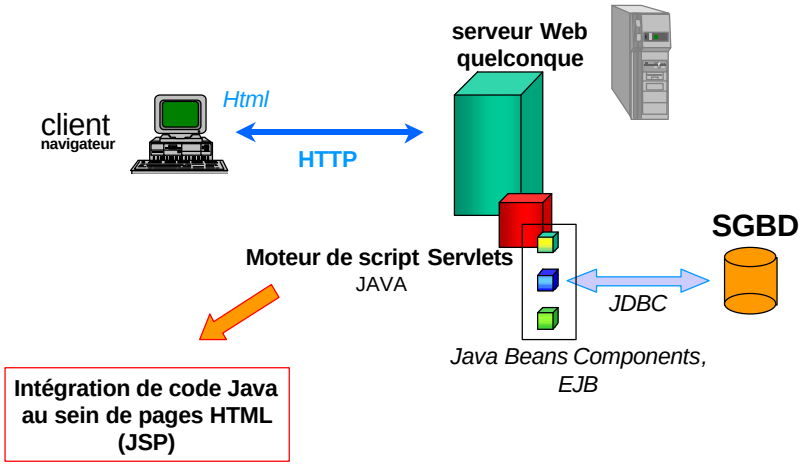
- Les *applets* sont généralement exécutées sur le poste de travail. Elles peuvent êtres localisées sur le poste mais sont la plupart du temps téléchargées à partir d'un serveur web de composants et exécutées dans le navigateur.



Source : Fi System

Figure 1 : Mode de fonctionnement d'une applet.

- Les *servlets* sont exécutés sur le serveur. Ce sont des composants soit conçus à des fins de présentation, soit à des fins de traitement. Les premiers sont appelés *Java Server Page* ou JSP et génèrent des pages web à destination de l'utilisateur, les second sont appelés *Java Beans*, ce sont des composants exclusivement utilisés en *back-office*.



Source : Fi System

Figure 2 : Mode de fonctionnement d'un servlet.

5. CLASSIFICATION

5.1 Portabilité

Le langage Java doit s'exécuter sur une machine virtuelle. Celle-ci est aujourd'hui disponible sur quasiment toutes les plates-formes, ce qui garantit en théorie une portabilité maximum.

En ce qui concerne les postes clients, les navigateurs les plus connus comme Netscape et Internet Explorer intègrent une JVM. En revanche, parmi les navigateurs moins répandus, certains exécutent du code Java mais ne l'intègrent pas en standard, et d'autres ne sont pas compatibles Java en natif. Néanmoins, il existe un composant additionnel Java fourni par Sun qui permet d'avoir les fonctionnalités Java et, qui plus est, d'avoir la même version sur des navigateurs différents.

Pour les serveurs de composants, si l'on respecte deux règles :

- développer en Java classique ;
- encapsuler systématiquement toute fonction propriétaire ;

alors on peut dire que la portabilité du langage est complète non seulement d'un environnement de développement à un autre, mais aussi d'une plate-forme technique à une autre (Java est « *cross-platform* »).

Certains membres du Cigref ont notamment constaté cette portabilité entre Visual Age for Java d'IBM et J-Developer d'Oracle.

Par le passé, l'évolution (trop) rapide des versions (3 versions majeures en 3 ans d'existence : 1. 0. 2, 1. 1. x, 1. 2/2. 0) ainsi que quelques pratiques singulières (Microsoft) ont pu poser des problèmes de compatibilité du code généré, mais ceux-ci sont aujourd'hui résolus. Il apparaît donc que la compatibilité ascendante est correcte.

5.2 Intégration technologique

L'architecture Java répond parfaitement aux standards de l'internet. Son évolution favorise son intégration avec des outils de génie logiciel et de *middleware*. Néanmoins, la cible principale de cette architecture s'est élargie au fil du temps :

- limitation à l'interactivité des pages web avec la version 1.0.x;
- extension aux applications sur poste client ;
- utilisation pour les serveurs et architecture multicouches.

6. PRINCIPAUX ACTEURS DU MARCHÉ

Pour l'architecture, selon SQLi, les *leaders* sont aujourd'hui Sun, IBM, Oracle et BEA. Microsoft fait aussi partie des acteurs en ce qui concerne la JVM intégrée à son navigateur mais son architecture ASP le met directement en concurrence avec l'architecture Java.

Pour le langage, le choix à faire n'est pas seulement entre Java et un autre langage objet mais aussi entre les éditeurs d'un même langage. Le critère de compétence technique de l'éditeur n'est pas le seul à prendre en compte, il faut aussi tenir compte de la taille de l'entreprise et de sa pérennité financière.

Une des difficultés réside dans la multiplication des rachats opérés entre éditeurs, ce qui a tendance à limiter le choix des entreprises.

6.1 Sun

Sun est à l'origine de Java et conserve un rôle fédérateur fort. Sa stratégie consiste à développer un maximum de partenariats avec des éditeurs et des constructeurs du monde de l'informatique, des télécoms et de l'audiovisuel (Nokia, Ericsson, Psion, Alcatel, Silicon Graphics, Gemplus, Schlumberger…) pour développer des API métier et rendre Java incontournable et le plus ouvert possible[1].

Parallèlement, Sun a entrepris également de racheter un certain nombre d'entreprises telles que Stardivision (suite bureautique Star Office), Netdynamics (serveurs applicatifs), Forte, Integrity Arts (API Javacard).

Sun cherche aussi à évangéliser le monde des développeurs. Le programme de certification de Sun, baptisé *« Sun Developer Connection »* compte 1,2 millions de personnes dans le monde, dont 40 000 en France. Par ailleurs, Sun cherche à simplifier son modèle de gestion des licences. Vendue auparavant entre 50 et 150 000 dollars la licence, Sun s'est rendu compte que ce mode de licences s'avérait trop coûteux pour les petits développeurs. Désormais, le téléchargement du code source est gratuit sur le site de Sun et le constructeur se rémunère sur un pourcentage des ventes réalisées par le partenaire. Auparavant, Sun comptait 200 sociétés licenciées. Aujourd'hui, ce nombre se situe entre 2 000 et 5 000 licenciés.

[1] Revers de la médaille, les API évoluent très vite et les développeurs ont parfois du mal à suivre.

Le mode de fonctionnement entre Sun et le développeur est le suivant : soit le développeur accepte le code fourni par Sun et ne le modifie pas, soit le développeur introduit une modification dans le code source et dans ce cas il doit passer par une phase de certification supplémentaire baptisée *« Java Specification Request »* (JSR). Cette phase de validation qui comprend à la fois des spécifications et des tests se décompose en 11 étapes, et dure 8 mois maximum. Il y a aujourd'hui 32 *Java Specification Request*, dans plusieurs domaines (téléphonie, temps réel…). Le propriétaire du code source est soit le développeur soit Sun.

Plus récemment, Sun a décidé de s'orienter vers une distribution en ligne gratuite des outils de développement et de sa suite bureautique Star Office. Cette distribution gratuite associée à une compatibilité entre Windows et Star Office a pour but de déstabiliser Microsoft, qui tire encore l'essentiel de ses revenus de la vente de licences. Cette distribution pourrait se faire notamment par le biais des ASP *(Applications Services Providers)*, des fournisseurs d'accès actuels qui se repositionneraient vers de la fourniture de contenu à la demande. Sun pense que l'utilisateur doit maintenant payer le service et non plus le produit. En conséquence, Sun va mettre en ligne l'équivalent fonctionnel de Star Office accessible *via* un portail de services Star Portal : ne sera téléchargeable que la fonction utile à l'utilisateur, la personnalisation des usages est alors possible mais devient payante.

Jusqu'à il y a peu, on pouvait penser que Sun cherchait à garder un contrôle sur la standardisation de Java, en témoigne sa sortie de l'ISO et tout récemment de l'ECMA *(European Computer Manufacturer Association)*.

Néanmoins, suite aux inquiétudes manifestes de la communauté des développeurs et ayant pris conscience de l'importance de l'indépendance de Java, Sun vient de modifier les règles de nomination des instances qui gèrent Java : le JCP 2.0 *(Java Community Process)* sera revu dans sa structure et surtout beaucoup plus ouvert : cette ouverture serait-elle un moyen pour Sun d'entrer de nouveau dans les instances de normalisation tout en gardant la tête haute ?

6.2 Oracle

Oracle, *« the 300 % Java company »* souffrait jusqu'il y a peu d'un manque de cohérence dans son offre. Les membres du Cigref ne comprenaient pas le positionnement d'Oracle 8i par rapport à Oracle Application Server (OAS). Pourquoi Oracle 8i intégrait une JVM dans le moteur lui-même alors qu'OAS utilise une JVM

externe (celle d'Inprise en l'occurrence). En plus, les utilisateurs reprochaient à OAS d'être encore trop propriétaire.

Plutôt que d'être 300 % Java, Il aurait mieux valu être 100 % cohérent Java.

Oracle vient de réagir en homogénéisant complètement son offre, OAS par un travail de migration devient IAS (Internet Application Server) et est 100 % compatible Java. L'élément le plus important est la JVM qui est celle utilisée par Oracle 8i (8.1.7) et qui intègre la compilation Java. L'ancienne JVM d'OAS n'est plus vendue, elle est figée en version 4.08.2 et maintenue en tant que telle.

Le package IAS se compose :

- du serveur web Apache ;
- du moteur Oracle IAS ;
- d'Oracle iCache qui est un cache de données vers Oracle 8i ; à terme, l'objectif est aussi d'être cache d'objets ;
- d'un ensemble de modules correspondant aux connecteurs d'OAS. Actuellement, nous pouvons notamment trouver MOD Wap (pour l'offre Portal to go) et MOD Java qui permet d'utiliser la JVM se trouvant sur le serveur Oracle 8i et de faire exécuter des traitements au niveau de ce dernier.

En ce qui concerne le langage, l'environnement de développement JDeveloper est apprécié pour son côté convivial et sa déclinaison dans de multiples architectures comme Unix, NT, Linux, S390... De plus, pour s'extraire en douceur de l'environnement propriétaire client-serveur d'Oracle en prenant la vague Java, JDeveloper est une bonne solution transitoire.

En septembre 2000, Oracle intégrera la compilation Java au produit. Cette compilation étant une transcription en C puis compilation C.

Pour 2001, un outil de conception UML viendra compléter le tout.

Enfin, le département Recherche & Développement d'Oracle avait mis au point pour ses besoins propres un *framework* (environnement de travail) pour les composants : il sera commercialisé sous le nom de Oracle Business Composant for Java (OBC4J). Ce *framework* permettra l'accès à des composants techniques et sera intégré à l'offre de JDeveloper.

6.3 IBM

IBM fait partie des entreprises les plus impliquées dans Java. En effet, Java est intégré dans l'ensemble de l'offre du constructeur, nommée webSphere.

Concernant le développement, les deux produits phare sont Websphere Studio et Visual Age for Java. Le premier est orienté développement de composants, le second développement d'applications. Visual Age se décline aussi en Visual Age for Packbase qui permet d'extraire des services Cobol et de les transformer en services Java et Visual Age application Roles qui, en incorporant le produit Versata, permet de développer des EJB sans maîtrise importante des objets Java une description des règles de haut niveau suffit pour entraîner un développement automatique d'EJB.

Pour les composants, IBM propose San Francisco qui est un *framework* de composants métiers multisecteurs. Il fournit à la fois les classes des objets Java et les composants. Mais IBM propose aussi Websphere eComponents qui regroupe à la fois San Francisco, CBTF (*Core Banking Teller Framework*) et Visual Component.

IBM propose aussi Websphere eCommerce suite intégrant un serveur Websphere, une JVM et des outils de développement.

Pour le *middleware*, il y a Websphere Application Server qui est édité en plusieurs éditions :

- *standard* (*servlet* et JSP) ;
- *advanced* (*Standard* avec les conteneurs EJB) ;
- *enterprise* (*Advanced* avec Tx Series et Component Broker).

De plus, IBM propose une série de connecteurs vers DB/2, CICS, MS/Series, IMS…

Enfin, l'offre *groupware* d'IBM s'articule autour de Domino Notes qui intègre le développement des agents Notes en Java, et dont tous les objets internes sont accessibles en Java.

Pour compléter cette offre, IBM propose ePortal qui s'articule autour de Domino Raven qui permet de faire de la gestion des connaissances et de profils d'utilisateurs et d'un portail websphere EIP *(Enterprise Information Portal)*. Ce dernier peut fournir des accès vers des serveurs web externes, des bases relationnelles, des serveurs LDAP…

Les principaux partenaires d'IBM pour cette offre sont Versata pour le moteur de développement, Versant pour les containers de persistance base de données objet, Ariba pour le relations interentreprises et BMC pour les agents d'administration Patrol for websphere. Prolifix fournit un pont entre websphere et Tuxedo et Rational un outil de modélisation (Rose) et un outil de gestion de versions (Clear Case).

Pour accompagner la vague Java, IBM a mis en place une plate-forme de services, Object Technology Group qui intègre des experts Java ainsi que d'autres technologies comme XML et Wap ; IBM Global Services fournit les forces vives de développement Java.

Pour les développeurs, IBM met à disposition des forums et des conférences web (ww.ibm.com/developper).

Enfin, Java mobilise 5 laboratoires soit 700 personnes auxquels s'ajoutent 3 000 développeurs.

6.4 Microsoft

Les début de Java n'ont pas été encouragés par Microsoft qui a essayé de mettre en avant son langage Visual Basic comme langage de *scripting* pour les pages interactives sur internet, l'opposant ainsi directement à Javascript. Le succès de ce dernier et le fait qu'il soit devenu un standard a conduit Microsoft à changer de stratégie et à se convertir à Java.

Mais très rapidement, Microsoft est accusé de « dévoyer » Java en proposant des fonctions qui sortent du standard et nuisent à sa portabilité.

Néanmoins, Microsoft possède une des JVM les plus performantes du marché sur le navigateur Internet Explorer.

Microsoft propose un atelier de développement intégré à la suite Visual Studio : Visual J++, et qui est jugé très intéressant à la condition de privilégier les bibliothèques et extensions non-Microsoft.

La nouvelle version de la suite Visual Studio ne devrait néanmoins pas incorporer dans un premier temps Visual J++, Microsoft mettant en avant un nouveau langage : C#, concurrent direct de Java.

6.5 Et les autres

Pour les *EJB :*

- BEA webLogic
- Persistence PowerTier for EJB
- Gemstone/J
- Etc.

Pour les *JDK :*

- BlackDown
- Inprise/Borland JBuilder
- Symantec Visual Cafe
- VisionSoftware VisionJade
- Progress Apptivity
- SilverStream
- Etc.

Pour les JDK autour de la *Javacard :*

- Odissey Lab (Bull),
- GemXpresso (Gemplus),
- GenerIC (Oberthur Card System),
- Cyberflex Open (Schlumberger)

7. CONTRAINTES LIÉES À JAVA

Java nécessite avant tout chose une bonne appréhension de la notion d'objet. Les développeurs C++ seront certes favorisés, mais il faut penser aux autres (Visual Basic, cobol, C, etc.). Un effort particulier doit donc être fourni pour passer des langages procéduraux aux langages objet. Il peut être intéressant dans certains cas de prévoir une étape C++ mais en règle générale, mieux vaut passer directement à Java.

Les contraintes se situent plus sur le poste client que sur le serveur. Si l'architecture Java simplifie les choses, une cohérence technique est indispensable ; en effet, il faut harmoniser les versions de JVM, prévoir une configuration matérielle adaptée, c'est-à-dire au moins 128 voire 256 Mo de mémoire vive pour les environnements de développement et les serveurs et 64 Mo sur le poste client. C'est-à-dire un environnement similaire à celui des applications bureautiques récentes. Enfin, il faut prévoir une bande passante suffisante pour télécharger les *applets* en cas d'utilisation de Java sur le poste client (cet usage tend à disparaître).

8. BÉNÉFICES LIÉS À JAVA

Indéniablement, tout le monde s'accorde à dire que la fiabilité et la stabilité de Java ne sont plus à prouver. Les cycles de développement sont raccourcis et les phases de tests allégées.

Java est maintenant enseigné dans les universités, écoles et instituts, ce qui contribue à sa démocratisation et à sa diffusion au sein de la communauté des développeurs.

Après la phase d'adoption, la productivité est supérieure par rapport aux autres langages, en raison de la réutilisation des objets. Java est maintenant un langage de développement universel et standardisé qui fait référence pour l'utilisation des technologies objet. Ces dernières le rendent particulièrement adapté aux développements sur des serveurs applicatifs avec la notion de composants objets ou d'objets métiers.

Les domaines d'application des architectures Java sont particulièrement vastes. D'autant plus que les performances sont en progression constante et que le catalogue des API disponibles s'enrichit continuellement.

Le marché est maintenant mûr et l'offre d'outils de développement est riche et sérieuse.

9. TYPES D'UTILISATIONS

L'utilisation la plus simple consiste en l'enrichissement des pages HTML pour des applications statiques et ne nécessitant pas de traitement. Java est alors soit intégré aux pages HTML, soit téléchargé sous forme d'*applets*.

La mise en œuvre d'applications lourdes ou métiers nécessite par contre des développements qui ne sont plus compatibles avec la configuration des postes clients et contraint à une exécution des applications sur les serveurs.

Dans ce cadre, l'architecture de référence Java est une architecture à trois couches : poste client, serveur de composants, serveur d'applications. Mais il est aussi possible de développer des applications Java dans une architecture à deux couches en mettant en œuvre des applications auxquelles le navigateur accède directement. Ce sont généralement des applications qui viennent en complément d'un serveur HTTP en utilisant soit des *servlets* (les pages HTML sont construites dynamiquement par l'application) soit des JSP (les pages sont statiques et intègrent du code Java qui dialogue avec l'application). Mais dans ces deux cas, les requêtes sont bien signifiées par le poste client directement à l'application.

On trouve aussi des architectures à deux couches qui mettent généralement en œuvre des applications intégrant à la fois le serveur de composants et la partie applicative. On se trouve alors avec deux étages sur le serveur : c'est une fausse architecture à trois couches.

Dans les architectures à deux et trois couches, la tendance actuelle est d'utiliser des EJB sous forme de composants métiers réutilisables.

En environnement extranet et internet, trois types de problèmes apparaissent : la compatibilité entre navigateurs, le temps de chargement des *applets* et les performances du poste client ; dans ces cas, on privilégiera des développements applicatifs intégrant des serveurs de composants ou des *servlets* qui construisent dynamiquement les pages HTML (d'autant que les applications sont de plus en plus développées pour des accès clients et fournisseurs). Par contre, l'utilisation d'*applets* est optimale au sein d'un intranet, pourvu que l'entreprise ait des normes internes pour les configurations de ses diverses entités.

Exemple d'*applet* :

- le Projet Atride chez EDF. Il s'agit d'un frontal graphique sous forme d'*applet* pour un système de gestion des données territoriales (SGDT) utilisé par des ingénieurs non-informaticiens (100 utilisateurs).

Exemples de *servlets* :

- Le site de bourse en ligne CPR E*Trade : projet comprenant un *servlet* côté serveur, un lien Corba mais pas de moniteur transactionnel ;
- Projet Savoir chez France Télécom : application pour les travaux de voirie regroupant 2 000 utilisateurs et générant 15 accès par seconde en moyenne. L'application s'appuie sur un moniteur transactionnel Tuxedo, une base de données Oracle et un serveur d'application IBM websphere ;
- Projet de *hot line* Itineris Digit 2G chez France Télécom Mobile : permet à l'appelant d'obtenir des renseignements sur les fonctionnalités de son GSM. Il a nécessité le développement d'une application HTML sur le poste client et d'une *applet* Java pour l'administration (saisie de nouvelles fiches).

Hormis les applications web, on commence aussi à trouver des applications sur des équipements embarqués comme les cartes à puce, les terminaux de paiement électroniques, les téléphones mobiles. Le principal attrait étant la possibilité de télécharger simplement de nouvelles versions et applications de manière transparente pour l'utilisateur.

10. COÛTS POUR L'ENTREPRISE

La comparaison doit se faire entre langages comparables (Java *vs* Visual Basic ou C++) et architectures comparables (Java *vs* Microsoft). De plus, elle n'est envisageable que sur un cycle de vie complet (3 ans).

Les technologies Microsoft (Visual Basic ou architecture ASP) semblent plus adaptées pour des petits projets et des développements au fil de l'eau, tandis que Java semble mieux conçu pour des grands projets structurés. Plus précisément, dans les organisations décentralisées où les développement se font de manière individuelle, Visual Basic est plus adapté ; mais la tendance actuelle est la recentralisation et donc le développement d'applications plus importantes avec une méthode globale et dans ce cas, Java convient mieux.

Concernant le volume de code, JSP et ASP sont équivalents.

Il est évident qu'un nouveau langage (Java) est dans un premier temps plus coûteux à adopter par une entreprise (migration, administration, formation...) qu'un ancien langage (Cobol, Visual Basic, C, C++) mais les bénéfices à moyen terme peuvent être supérieurs.

Le coût tend même devenir à devenir secondaire lorsque le projet est considéré comme stratégique par l'entreprise et la direction générale (les nouveaux projets tels que l'*e-business* par exemple).

Si l'on détaille chaque coût, on note plusieurs points.

- Coûts de développement : Java nécessite un redéveloppement initial obligatoire des applications de l'entreprise. Le coût de développement en java peut être 15 % plus cher que sous Visual Basic ou C (tarifs des SSII). Mais il faut jouer à fond la carte de la portabilité future.

  Il faut aussi noter que si la station de développement Visual Basic peut être la machine de monsieur tout le monde, celle pour développer sous Java doit être beaucoup plus puissante.
- Coûts de migration : une migration vers une architecture de type intranet ou internet doit être envisagée si l'on passe d'une architecture classique à une architecture Java. Mais attention, si l'architecture est homogène et basée sur Microsoft NT, alors Java n'offre pas plus de facilités que l'architecture Microsoft ASP.
- Coût de déploiement : le coût de déploiement en environnement intranet est plus faible qu'en environnement

client-serveur propriétaire, ne serait-ce que parce qu'il n'y a pas d'application à installer sur le poste client.

Néanmoins, les serveurs d'applications sont plus chers, puisque selon les membres du Cigref, il faut compter de l'ordre de 50 kF pour un monoprocesseur et 60 kF par processeur supplémentaire.

- Coûts d'administration : ils sont plus faibles que pour les architectures client-serveur. Il y a en l'occurrence moins d'administration client. Néanmoins, les membres du Cigref estiment qu'il faut environ 30 jours de formation pour les administrateurs.
- Coûts de formation : nécessité d'une compréhension de la méthode de conception objet par les développeurs traditionnels (Cobol, Cics, SGBDR...). Mais la multiplication des formations initiales et professionnelles en Java va entraîner une baisse des coûts de formation et une diffusion de Java dans le monde des développeurs.

  Il est à noter qu'à compétences initiales égales les membres du Cigref considèrent que le coût de formation Java est identique à celui de Visual Basic ou de C/C + + ;

  Selon le CXP, il faut compter de l'ordre de 20 kF par licence de développement Java, plus 25 kF de formation par développeur. Pour une équipe de 20 développeurs sur 5 ans, on en arrive à des coûts de formation de 4 MF pour Oracle et de 5,5 MF pour IBM ;
- Coûts de maintenance : Dans le cas des architectures multi-couches, il y a report du coût des postes de travail sur les serveurs, mais ce n'est pas lié à Java.

11. BÉNÉFICES POUR L'ENTREPRISE

Il apparaît que la fiabilité de Java et ses facilités de développement entraînent, selon les membres du Cigref, un raccourcissement de la durée des projets informatiques (de 1 an en moyenne à 6 mois). Cela va sans doute conduire à une adoption massive de Java permettant notamment d'introduire des composants réutilisables.

On peut aussi penser que la réutilisabilité et la portabilité des applications développées en Java vont permettre de simplifier grandement les migrations d'applications d'un système d'information vers un autre, notamment lors des fusions ou acquisitions : Java favorise l'indépendance technique des outils métiers de l'entreprise.

De l'avis des développeurs membres du Cigref, le niveau d'abstraction supplémentaire que le langage Java apporte par rapport à C++ permet d'envisager de façon plus sereine le développement d'objets métiers et plus généralement des développements pérennes. D'autant plus que, hormis les API propriétaires encapsulées, les contraintes de développement en Java sont plus souples que dans les autres langages.

Concernant l'architecture, le bénéfice le plus important par rapport à une architecture classique client-serveur est de ne pas avoir de déploiement applicatif sur les postes, d'où une simplification de la mise en œuvre sur le serveur. Les architectures à trois couches permettent ainsi d'éviter d'avoir un *middleware* entre le poste client et le serveur et les problèmes de configuration et de version qui en découlent. (Les *middleware* peuvent néanmoins être reportés sur les serveurs de composants ou applicatifs, le paramétrage est alors plus centralisé.) Les incidences directes en sont notamment un allègement de la charge des *help desks* et hotlines qui peuvent se concentrer sur une aide à l'utilisation et non au bon fonctionnement voire à la bonne configuration.

Vis-à-vis des fournisseurs, la standardisation assure une relative indépendance, à la condition de respecter un dénominateur commun (ou d'encapsuler ce qui est spécifique). Et il y a actuellement une forte attente des entreprises sur les objets métiers notamment vis-à-vis des progiciels de gestion intégrés (SAP, Intentia…).

12. ÉVOLUTION PRÉVISIBLE

12.1 Court terme

Le langage Java ne va pas remplacer les langages classiques comme Visual Basic et C qui répondent à des besoins spécifiques de développement rapide pour des petites applications (Visual Basic) ou performants parce que proches du système (C) ; Le C++, par contre, n'a plus de raison d'être.

Sur les serveurs, la substitution se fera de manière douce : des applications côté serveur (*servlets* et JSP) vont se substituer aux CGI et autres ASP dans un premier temps. Selon le Gartner Group, les serveurs d'application EJB rattraperont leur retard d'ici 2004 puisqu'ils prévoient un marché à 50 % de serveurs MTS (Microsoft) et à 50 % de serveurs EJB.

Les compétences, elles, arrivent, ne serait-ce que parce que le langage Java est aujourd'hui enseigné actuellement systématiquement dans les universités.

12.2 Long terme

La technologie objet a maintenant les bases pour s'imposer : les objets métiers s'imposeront à terme d'une part avec l'arrivée d'outils évolués (à base de règles métiers) intégrant un ORB et des connexions vers des bases de données et des ERP ; d'autre part avec l'émergence à terme de composants métiers sur étagère.

Quel sera l'impact de Java sur les postes clients ? le langage Java et l'architecture associée vont-ils faire évoluer des services comme la messagerie, les annuaires et plus généralement les outils bureautiques ? L'annonce de Starportal (offre d'outils bureautiques à la demande) faite par Sun, paraît aller dans ce sens.

13. COMMENTAIRES

Java permet de retrouver une approche professionnelle du génie logiciel (cycle de vie, qualité et test, outillage, intégration), avec de nouvelles façons de voir (orientation objet, démarche itérative).

L'architecture Java s'oppose à l'architecture Microsoft et il n'y a plus d'autre alternative. Le passage de l'une à l'autre peut induire une forte inertie au changement, certes naturelle, de la part des anciens développeurs.

L'évolution des environnements Java est très rapide par rapport au rythme beaucoup plus lent des évolutions technologiques dans les grandes entreprises. Par comparaison, l'évolution des développements en C++ a été 3 fois plus lente qu'en Java.

14. RECOMMANDATION

De par leur évolution très rapide, les technologies multicouches sont encore jeunes et n'ont pas complètement fait leurs preuves. Il faut donc raison garder et ne pas céder à l'effet de mode en passant d'un coup son système d'information en intranet.

Les deux aspects à surveiller sont notamment la stabilité et la capacité de s'adapter au changement d'échelle (*scalability*) : si Java est une plate-forme mûre, sa stabilité, qui s'améliore notablement avec Java2, n'est pas encore totale ; Si la *scalability* paraît être un des point forts, elle mérite une attention particulière due à la jeunesse de l'architecture Java.

Enfin, les temps de réponse induits par l'architecture Java la destine, pour l'instant, à des applications intranet et non internet.

ANNEXE : Lexique

API : *Application Programing Interface*. Ensemble de fonctions regroupées dans une bibliothèque et permettant d'accéder à un composant particulier.

Applet : application téléchargée depuis un serveur web qui s'exécute dans une JVM intégrée au navigateur.

EJB : *Enterprise Java Bean*. Composant métier exclusivement utilisé sur les serveurs : il nécessite un serveur dédié. On appelle ce composant *via* un gestionnaire de Corba, un *Servlet* ou encore RMI.

ERP : *Enterprise Resource Planning*. Progiciel de gestion intégré.

JAAS : *Java Authentication and Authorization Services*.

JAIN : *Java Advanced Intelligent Network*.

JCE : *Java Cryptography Extensions*.

JDK : *Java Development Kit*. Environnement de développement Java permettant de produire du code Java et servant de référence.

JINI : permet de connecter des périphériques au réseau de manière « plug and play ».

JNI : *Java Native Interface.* Permet l'interopérabilité de Java avec d'autres langages en offrant l'accès à des machines virtuelles.

JSSE : *Java Secure Socket Extensions*.

JVM : *Java Virtual Machine.* La machine virtuelle de Java est à la base même de sa portabilité sur la plupart des plates-formes. En réalité, le code Java est compilé en instructions compréhensibles par la machine virtuelle qui elle-même interprétera ces instructions pour que l'ordinateur puisse les comprendre. C'est pour cela que chaque plate-forme doit disposer de sa propre machine virtuelle pour fonctionner

ORB : *Object Request Broker*. Dans un environnement à objets, programme régulant les échanges de messages entre les objets

RMI : *Remote Method Invocation*. Invocation de méthode distante, mécanisme permettant d'utiliser sous Java des objets distribués.

Servlet : application s'exécutant sur le serveur et générant des pages HTML.