AJAX pour EIAH ?

Denis Bouhineau

AJAX pour EIAH ?

Choix technologique pour un EIAH de l'algorithmique : EDBA

Denis Bouhineau*

** Laboratoire d'Informatique de Grenoble (LIG)*
Université de Grenoble (Grenoble-I, Univ. J. Fourier)
Denis.Bouhineau@imag.fr

RÉSUMÉ. *À l'occasion de la réalisation d'un projet d'EIAH de l'algorithmique (EDBA : Exercises DataBase about Algorithms), le choix de la technologie support de développement s'est tourné vers AJAX, plus particulièrement vers une version d'AJAX quasi-uniquement à base de Javascript, donnant à l'EIAH visé l'aspect d'un client lourd intelligent permettant des déploiements sous forme d'application web mono-page s'intégrant facilement à des plateformes d'enseignement type Moodle. Les raisons de ce choix, les attentes et interrogations a priori, sont exposées ainsi que les réponses issues des développements d'une première version de l'application cible. À travers cette expérience, la question de l'indépendance de la recherche vis-à-vis de la technologie est aussi évoquée. Quelques retours d'utilisation in vivo de l'application cible par des étudiants volontaires complètent la proposition pour une validation par l'exemple.*

MOTS-CLÉS : *EIAH, Algorithmique, Programmation, Interpréteur, EDBA, AJAX, Javascript, Php, MySql, Architecture, Développement logiciel, Déploiement, Plateforme pédagogique, Moodle, Base de données, Base d'exercices, Prolog, CAML, Diagnostic, Rétroaction, Collaboratif.*

Introduction

L'engouement pour AJAX, Asynchronous Javascript and Xml [Garrett – 05], c'est-à-dire pour les applications s'exécutant dans les navigateurs, sans installation d'éléments supplémentaires (plugin), ni installation sur l'ordinateur d'applications spécifiques, semble ne pas devoir s'arrêter depuis son lancement en 2005. Prenant son origine avant l'invention du terme AJAX, avec l'introduction 10 ans auparavant, en 1995, dans les navigateurs, d'un langage de programmation -Javascript (normalisé sous le nom d'Ecmascript, avec des variantes, jscript, actionscript, par exemple)- permettant l'écriture de programmes interprétés et exécutés par le navigateur lui-même. Le mouvement AJAX a suivi de peu l'ajout d'un objet logiciel dans les navigateurs, XMLHttpRequest, en 1998, 2002, 2005, selon les navigateurs pour accéder par programme aux données accessibles sur internet. Sur la période 1995-2010, la popularité de cette technologie n'a cessé de croître comme le montrent les diagrammes donnés par Google-Chronologie. La croissance d'Ajax suit celle, similaire, de Javascript ; selon une autre source, le maximum de la popularité de Javascript a été observé en Juin 2009 par l'éditeur de logiciels TIOBE lors de ses études sur la popularité des langages de programmation [TIOBE]. Pour comparer, la popularité de Java semble beaucoup plus stable sur la même période (d'autres exemples peuvent être pris avec Php, ou Perl). L'utilisation courante de cette technologie par Google et le soutien que ce géant lui apporte peuvent expliquer une part de ce succès. À en croire certains prophètes, il semble même que les applications telles qu'on les connait depuis 20 ans, dites applications de

bureau (desktop application), soient vouées à disparaitre (fsmsh.com/2940), l'avenir étant à la programmation d'application web ne nécessitant qu'un navigateur pour s'exécuter. Sans pour autant souscrire à ces prophéties, il est clair que nombre d'utilisations de l'informatique passent aujourd'hui par le web, en excluant l'accès à Wikipédia ou aux diverses informations disponibles sur internet et qui constituaient initialement le web (tout le web !). Citons entre autres (avec quelques exemples) : l'accès à sa messagerie (via webmail) ; la gestion courante des bases de données (via PhpMySqlAdmin) ; la mise à jour des sites web (via Spip) ; la prise de rendez-vous (via Doodle) ; l'élaboration d'enquêtes (via Sphinx) ainsi que leur passation et l'analyse des réponses ; l'organisation pédagogique des cours (via Moodle) ainsi que la réception et la correction des copies ; etc. *Certaines journées de travail semblent ne pas devoir quitter le navigateur de notre ordinateur ! Nombre de ces applications passent par AJAX et les technologies du web 2.0*, cf. [O'Reilly – 05].

Le domaine des EIAH est peut-être à la pointe de la technologie en informatique (les technologies numériques de la communication ou les technologies éducatives ne sont pas l'objet de cet article, le terme *technologie* signifiera ici et dans la suite *technologie informatique*, sous entendue pour le développement d'EIAH). Si l'on considère AJAX, il semble au contraire que cette proposition n'ait que peu séduit la communauté des chercheurs et enseignants. La typologie des applications web reposant sur AJAX, avec surtout du contenu à lire, une bonne capacité à communiquer via internet pour acquérir les données nécessaires mais une interactivité réduite, sommaire vis-à-vis de l'utilisateur et peu d'intelligence, est peut-être l'une des raisons de ce manque d'utilisation. La question des technologies informatiques supports des mises en œuvre n'est pas souvent abordée dans les articles, citons comme exception [Trgalova et al. – 09], aussi il est difficile d'affirmer avec certitude qu'AJAX n'est pas utilisé dans certains EIAH. Sur la période 2003-2010, la revue Sticef a publié un peu plus de 60 articles, si l'on se réfère aux titres aucun ne concernait explicitement la technologie et moins de 5 abordaient le sujet dans le résumé, dont en particulier [Rebai et al. – 05] et quelques rares autres articles du dossier spécial *Conceptions et usages des plates-formes de formation*. *La technologie, les termes « Java », « XML », « PHP », sont-ils des mots tabous qui n'ont pas leur place dans les titres et résumés ? Le choix d'une technologie n'est-il pas important ? Toutes les technologies sont-elles équivalentes ?* Pour prendre quelques exemples personnels, il me semble au contraire qu'il y aurait matière à disserter, en considérant seulement les choix technologiques, sur ces premiers projets auxquels j'ai participé, entre "GéoSpécif" [Bouhineau – 97], sous mac-OS, nécessitant un interpréteur spécifique (PrologIII, payant de surcroît, quasiment disparu depuis) et les travaux entrepris ensuite pour "Aplusix" [Bouhineau – 03] pour Windows, sous forme d'exécutable autonome distribué librement un temps sur le web avec installateur et documentation, puis avec une distribution commerciale (gratuite pour la recherche).

À l'occasion de la mise en place d'un nouvel EIAH pour l'algorithmique (EDBA : Exercises DataBase about Algorithms, [EDBA]) en 2008, une réflexion sur le choix de la technologie qui allait servir de support aux développements de l'EIAH a été entreprise. Au lieu d'être considérée comme une contrainte, ou d'être prise comme un mal nécessaire, il a été décidé que la technologie support de développement serait l'un des moteurs de la réflexion et de l'innovation. Outre la technologie, le projet EDBA a des objectifs pédagogiques tournés vers les apprenants : introduction de la notion de test/jeux d'essai dans l'apprentissage de l'algorithmique comme mode de validation et élément d'apprentissage, de

suivi et de motivation des apprenants (en faisant un lien entre la réussite lors de la résolution d'exercice, la maîtrise algorithmique des apprenants et la difficulté des exercices, et en accordant aux apprenants des droits pour la rédaction des exercices semblables à celles que l'on donne habituellement aux enseignants). D'autres objectifs étaient plus tournés vers les enseignants ou les chercheurs en didactique de l'algorithmique : accumulation et structuration de matériel pédagogique, exercices d'algorithmique, sous la forme d'une base d'exercices, de jeux d'essai, d'aides, d'exemples, de tentatives de résolutions réelles par des étudiants, etc. empruntant une démarche participative (enrichie par les utilisateurs, soit explicitement pour les experts rédacteurs d'exercices, soit implicitement pour les étudiant lors de leur utilisation de l'EIAH). Relativement à la technologie, le projet avait pour objectif d'explorer les technologies AJAX, de choisir la/les variante(s) d'AJAX les plus appropriées, pour étudier dans quelle mesure elles pouvaient être utilisées dans le domaine des EIAH et si elles comportaient des limitations ou sous quelles conditions les spécificités des applications EIAH peuvent être supportées par la technologie AJAX. Les raisons de ce choix a priori seront exposés dans la première partie de cet article avec les attentes et les questions posées à la technologie. La seconde partie fera un retour sur les solutions et succès obtenus en utilisant cette technologie lors de la réalisation d'une première version de l'EIAH visé. La dernière partie se fera l'écho des premiers usages de l'EIAH pour fournir une forme de validation de l'utilisation de cette technologie pour la réalisation d'un EIAH par des preuves d'utilisation in vivo.

1. Attentes et questions vis à vis de la technologie AJAX

1.1. Intégration d'EIAH dans les plateformes pédagogiques ?

Le choix initial d'AJAX avait pour contexte le mouvement d'informatisation, de numérisation et de mise à disposition via internet des enseignements qui a vu le jour avec la diffusion des plateformes pédagogiques d'enseignement type Moodle (d'autres CMS pédagogiques, ou LMS, peuvent être cités : Sakai, Dokéos, …), cf. [Graf & List – 05]. L'apport de ces plateformes est clair ; cependant le lien en ces plateformes et les EIAH est loin d'être évident. Prévues pour être génériques, ces plateformes sont le plus souvent ignorantes des contenus et se veulent indépendantes des matières enseignées : ces plateformes supposent donc que les matières seront abordées via des documents, éventuellement des QCM, des modules de travail collaboratif (wiki, news, chat, …), ou des renvois à d'autres environnements web via leur url ou d'autres applications étrangères à ces plateformes (c'est là que l'on peut retrouver des EIAH). L'intégration de contenus dans ces plateformes passe donc par le HTML, des url, ou des références externes (c'est-à-dire, sans réelle intégration, dans ce cas). HotPotatoes (hotpot.uvic.ca) l'a bien compris qui offre un producteur de pages HTML de QCM (et autres questionnaires similaires) à base de Javascript pouvant être utilisées dans Moodle. D'où une certaine pauvreté des contenus interactifs réellement intelligents effectivement présents dans ces plateformes et la nécessité, si un enseignant souhaite bénéficier des deux (plateforme et EIAH) de jongler entre plusieurs environnements. Sauf quelques rares exceptions, les EIAH, avec leurs applications lourdes nécessitant des installations n'ont, en effet, pas réussi à intégrer ces plateformes. La première question posée à la technologie sera donc : *Développer un EIAH en AJAX, c'est-à-*

dire avec le langage du web, comme une page web intelligente, permet-il une intégration dans les plateformes pédagogiques ?

1.2. Une application AJAX intelligente ?

La description rapide des applications AJAX données précédemment -comme des applications ayant surtout des capacités à interroger internet et diffuser des contenus à lire, regarder ou analyser, contenus qui s'y trouvent déjà ou qui sont produits à la demande, mais des applications ayant peu de capacité en soi pour produire des comportements riches, intelligents ou pédagogiques- peut inquiéter pour un projet d'EIAH d'algorithmique. Tout comme les EIAH en mathématique peuvent avoir besoin d'expertise fournie par des systèmes experts ou des logiciels de calcul formel, ou nécessiter une intelligence embarquée, l'enseignement de l'algorithmique nécessite des capacités d'analyse, d'exécution, d'édition, etc. des solutions algorithmiques proposées par les apprenants. Une solution peut toujours passer par des QCM, auquel cas les plateformes pédagogiques pourront répondre à la demande ; mais si la pédagogie employée nécessite un engagement plus important de l'apprenant, si la démarche d'enseignement est plus active et passe par la résolution effective de problème via la programmation d'une solution, l'EIAH produit en AJAX ne pourra pas faire l'économie d'une capacité à manipuler, comprendre ou interroger les programmes produits par les apprenants. La seconde question posée à la technologie sera donc : *La technologie AJAX permet-elle le développement d'applications suffisamment intelligentes pour prendre en compte des contenus complexes, permettre à l'utilisateur d'interagir avec, de les travailler, d'en obtenir un diagnostic ou des retours, rétroactions ou indicateurs épistémiques et d'arriver à des productions de complexité comparable à celle obtenue dans les EIAH produits avec les technologies habituelles ?*

1.3. La recherche, un monde ouvert, transparent ? L'enseignement, aussi ?

L'une des particularités étonnantes d'AJAX est consubstantielle à son origine : destinée à être au service de la diffusion de contenus. Quand AJAX repose sur Javascript -ou pour ce qui concerne la part d'AJAX qui repose sur Javascript- le code des applications non seulement s'exécute dans le navigateur sur l'ordinateur de l'utilisateur, mais il est de plus visible et modifiable. Ainsi, de fait, la part Javascript des applications AJAX, appartient au monde de l'open-source : les sources sont disponibles, l'utilisateur peut les lire, les adapter à son utilisation, les rediffuser, …

Une partie de l'activité de la recherche est caractérisée également par cette ouverture, de cette transparence du travail, pour que les résultats obtenus puissent être diffusés, vérifiés, prolongés. Cependant, l'essentiel de la diffusion de ce travail passe par la publication d'articles de recherche et ne comporte que rarement les programmes et les données analysées, in extenso. Ce serait impossible, certes, dans ces articles. Mais, hors article, la diffusion des programmes et données est loin d'être une évidence, sans pour autant que cela soit la preuve d'une volonté de préserver ses résultats, ou une avance technologique (au sens large) dans ses travaux de recherche. La plupart du temps, la diffusion des programmes et données n'a pas lieu, car les pré-requis logiciels, les problèmes de licences éventuelles, les contextes d'exécutions des programmes, rendent le plus souvent une diffusion des programmes « tel-que » inutile, parce qu'inopérante, vouée à l'échec. Et ce, sans pour autant

que cela signifie que les logiciels en question ne fonctionnent pas, ou que les données ne sont pas réelles. Le travail pour qu'une application puisse sortir du laboratoire et soit exploitable, partageable, utilisable par autrui est vraiment non négligeable. Aussi vient une question : *Avec un choix technologique comme AJAX, est-ce que la diffusion d'un programme sera plus simple ? (et la recherche plus ouverte ?)*

Notons qu'au niveau technique, pour que cela soit le cas, la variante d'AJAX à choisir doit reposer pour l'essentiel sur du code Javascript disponible dans les pages HTML diffusées. D'autres technologies associées à Ajax : PHP, Java-GWT, par exemple, conservent une part de code qui peut ne pas être diffusée.

Pour certains, cette transparence de la programmation AJAX est un danger pour l'enseignement, cf. [Rowe – 04], et vis-à-vis de la vulnérabilité des systèmes informatiques. Effectivement, donner le code de ses programmes permet de les analyser, et éventuellement d'aller voir, par exemple pour les QCM, assez facilement les réponses attendues sans effort (sauf un effort de lecture d'AJAX). De même, la publication de codes, peut engendrer une plus grande vulnérabilité des programmes face aux attaques de hackers, non pour tricher cette fois, mais pour atteindre et exploiter les LMS. Sans que cela soit une réponse définitive, une solution donnée par la communauté AJAX consiste à obfusquer le code (rendre le code impénétrable, illisible par l'homme, au moins) ; en existe-il d'autres pour l'EIAH ? *La transparence d'AJAX peut-elle être utilisée pour l'enseignement et les EIAH ?*

1.4. Des outils pour le développeur et l'administrateur ?

La typologie des applications AJAX initiée plus haut présentait ces dernières comme des applications peu intelligentes. L'une des raisons de cet état de fait vient de ce que nombre d'entre elles sont de *petites* applications, parfois vues même comme des gadgets. Ces petites applications tiennent souvent en quelques lignes ou pages de code Javascript, facilement lisibles, modifiables, échangeables. En approfondissant un peu l'analyse des plus *grosses* applications AJAX, on peut observer que la plupart d'entre-elles ne sont pas écrites nativement en Javascript, mais dans d'autres langages (Java le plus souvent, comme par exemple pour les applications utilisant GWT : Google Web Toolkit), les langages usuels de développement d'applications standards, langages pour lesquels des outils de développements et de mise au point ont été définis (Eclipse, Netbeans, …) Ironie du sort, pour Java, alors que le mécanisme des applets était prévu pour permettre aux applications java de s'exécuter dans les navigateurs, ces grosses applications sont rarement écrites pour fonctionner sous forme d'Applet, mais pour être compilées en Javascript et produire au final une application AJAX, Javascript devenant ainsi le langage d'assemblage des navigateurs web perçus comme plateforme d'exécution et système d'exploitation. Est-ce à dire qu'écrire en Javascript s'apparente à écrire en assembleur ? La question n'est pas si évidente. Au niveau du langage, lui-même, Javascript a non seulement la puissance formelle de n'importe quel langage (il est équivalent à une machine de Turing), mais c'est un langage moderne qui appartient à la catégorie des langages de haut niveau et non pas à celle des langages de script (malgré son nom), encore moins aux langages d'assemblage. C'est un langage orienté objet, en particulier.

La définition du langage n'est pas tout ; écrire un gros programme nécessite aussi des outils adaptés (EDI) et malheureusement pendant longtemps les outils disponibles pour le développement et la mise au point de programmes en Javascript ont été, en nombre et en qualité, très réduits. L'un entrainant l'autre, l'écriture d'applications AJAX -nécessitant de plus des compétences nombreuses (pour les aspects purement techniques : en HTML, en DOM, en http, en CSS, en Javascript aussi, et c'est sans compter avec les compétences en infographie, en ergonomie, en IHM)- a longtemps été considérée comme quelque chose de difficile voir d'impossible, d'autant moins réaliste que la taille des applications visées était importante. Aussi : *Est-ce qu'aujourd'hui les choses ont changé, est-il possible de développer une grosse application AJAX à base de Javascript natif ? Des outils de développement sont-ils disponibles, de bonne qualité ? Les bases d'une ingénierie logicielle sont-elles présentes ?*

Au-delà du développement, il faut aussi considérer la question du déploiement, de l'administration des données et de la mise à jour des applications. C'est un point fort revendiqué par AJAX : il semble qu'il suffise d'avoir un navigateur et internet pour que tout soit toujours disponible et à jour. Tout se passe à distance sur le serveur. Que cache cette vision idéale ? *Que se passe-t-il si le réseau est filtré ? ou lent ? ou même indisponible ? Faut-il avoir recours à des outils supplémentaires (coté serveur) ? Quel sera le surcoût pour le développement d'un EIAH de devoir gérer cette distance ? Et à propos des navigateurs, tous vont-il fonctionner ?*

2. Retour après une première réalisation : solutions, choix, limites.

Globalement, toutes les questions posées vont recevoir une réponse positive puisque nous avons réussi à obtenir une première version d'un EIAH réalisé avec AJAX, cf. figure 1. Cependant, un certain nombre de choix ou de solutions ont dû être faits ou trouvées. Certaines solutions sont partielles, certains choix introduisent des limites, … l'ensemble mérite quelques explications.

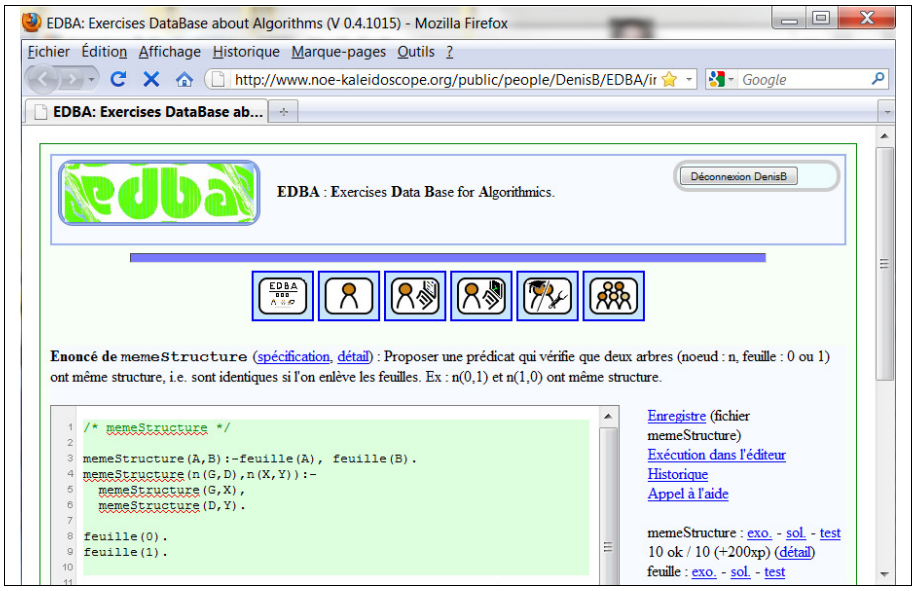

Figure 1. *Copie d'écran d'EDB. www.noe-kaleidoscope.org/public/people/DenisB/EDBA/index_EDBA_Full.html*

En quelques mots, l'application fournie permet à un apprenant de s'exercer en algorithmique selon un scénario proche du suivant :

- Choix d'un exercice, type rédaction d'un algorithme, dans une liste issue d'une BD, parmi ceux accessibles pour l'expertise de l'apprenant, en prenant en compte la difficulté des exercices, les notions abordées ou les noms d'exercices (si ceux-ci sont assez explicites) *ou* travail sur un exercice donné par un enseignant *ou* travail libre,
- Rédaction d'une solution pour l'exercice considéré sous la forme d'un programme dans un langage de programmation donné (pour l'instant Prolog et CAML sont disponibles, un mini-C ou Javascript est prévu dans l'avenir, un assembleur ARM est également prévu : 4 paradigmes de programmation seront alors pris en compte),
- Test de la correction de la solution candidate par exécution de celle-ci sur des jeux d'essai de référence et comparaison des résultats obtenus avec les résultats de référence (stockés dans la BD),
- En cas de succès, l'utilisateur (sauf les anonymes) gagne de l'expertise et peut travailler sur des exercices plus complexes,
- Pour les utilisateurs experts, possibilité d'introduction de nouveaux exercices et de jeux d'essai dans la BD.

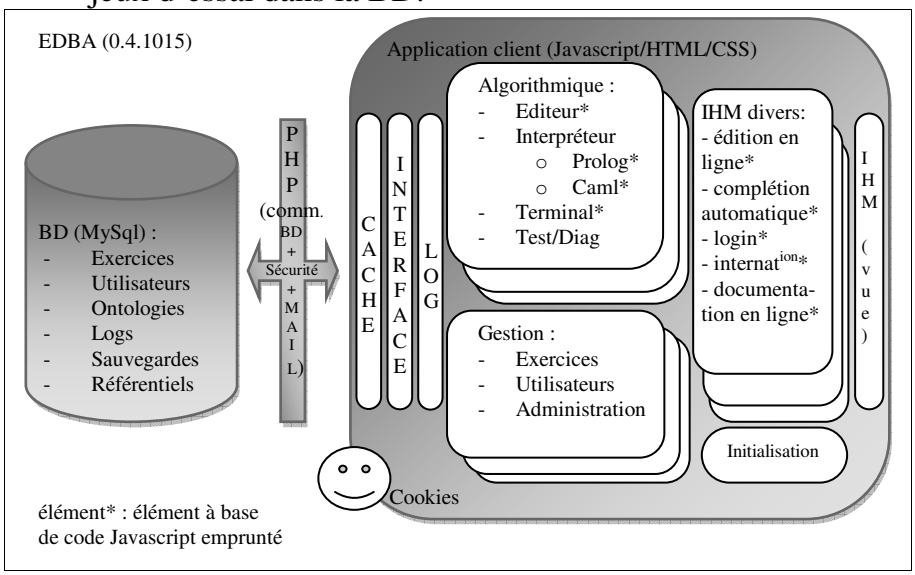

Figure 2. *Architecture globale d'EDBA.*

L'architecture adoptée pour EDBA est une application AJAX multi-tiers utilisant trois séries de technologies : une partie, téléchargée sur le poste client (Javascript+HTML/CSS), constitue l'essentiel du coté applicatif (IHM+métier) ; une minuscule partie (PHP, générique) fait le lien avec une troisième partie garantissant la persistance des données d'EDBA (en BD, via un serveur MySql) en même temps qu'elle assure quelques fonctions de sécurité et l'envoi de courriels. Voir la figure 2 pour un schéma global. La quasi-totalité de l'application se trouve donc au niveau applicatif écrit en Javascript ; les données, sur les exercices et sur les utilisateurs, se trouvent quant à elles, pour essentiel, en BD. Un système de cache limite les accès en lecture à la base de données et permet de travailler sous certaines conditions (cache rempli) hors connexion. Les accès en écriture (pour les logs), quant à eux, sont pour la plupart asynchrones ce qui permet de ne pas pénaliser l'utilisateur avec des temps d'attente liés aux aléas du réseau. L'essentiel de l'application est donc sur le poste client de l'utilisateur et, pour les parties critiques, seul l'ordinateur client fonctionne.

Le projet EDBA n'aurait peut-être pas été lancé si une recherche sur internet n'avait montré la disponibilité de codes, modules ou applications pour l'édition, l'exécution de code en Prolog, Caml (et dans quelques autres langages) et la simulation de terminaux comme les étudiants en informatique en utilisent pour tester leur programmes. Sur le diagramme donné à la figure 2, les éléments suivis d'une étoile (élément*) désignent certains de ces modules ainsi que les autres empruntés à internet et adaptés pour EDBA. Comme cette architecture le montre, une grande partie du code d'EDBA n'est pas originale. Ces modules assurent une partie non négligeable de l'intelligence de l'application. Les autres parties intelligentes (algorithme de diagnostic d'un exercice par comparaison sur des jeux d'essai de référence, et la donnée des exercices et jeux d'essai contenus en BD) sont ad'hoc ou ont été produites à l'aide d'EDBA (pour la BD), puisqu'EDBA permet l'extension de la BD interactivement.

2.1. Quelques réponses positives et réussites

Sur la réutilisation de codes, Javascript, facteur de développements logiciels rendu communautaires, du fait de la transparence du code, permettant la réalisation d'applications « intelligentes » reprenant les standards habituels : la possibilité de retrouver, d'assembler, de paramétrer des codes venus d'internet pour former un tout cohérent n'est pas pour autant évidente. Différents problèmes de captation de nom, d'utilisation de bibliothèques ou de technologie tiers que l'on ne désire pas, nécessitent parfois la réécriture de certaines parties incompatibles. Pour faciliter ce travail d'assemblage, lorsque le choix entre plusieurs modules réalisant la même fonction est possible, nous suggérons de choisir les modules les plus simples. Ces emprunts sont, pour autant, le plus souvent préférables à la réécriture des modules en questions. Outre les modules considérés dans EDBA (éditeur de code, terminal, interpréteur), d'autres modules 'intelligents' peuvent être considérés, par ex. pour des explorations statistiques de données, des représentations graphiques ; la définition et manipulation d'ontologies ; etc.

Sur les bases d'une ingénierie logicielle : l'existence de cette masse de codes séparés n'est cependant pas suffisante, elle signifie plusieurs centaines de ko de code à fusionner et à maintenir (près de 300 ko de codes empruntés et tout le code original d'EDBA – à peu près autant de ko). Des outils de génie logiciel sont nécessaires. Jusqu'en 2006-2007 ces outils étaient assez pauvres (pour Javascript) et imparfaits, ce qui constituait un réel verrou pour le développement de grosses applications AJAX à base de Javascript ; cependant sont apparus à peu près à la même période deux éditeurs de Javascript de qualité satisfaisante pour les EDI Eclipse et NetBeans, ainsi qu'un bon débugger : FireBug. Le développement d'applications lourdes en Javascript pouvait donc commencer. EDBA a commencé en 2008-2009 en tirant profit de cette opportunité. À ce jour, le panorama a peu changé. Notons l'existence, également, d'outils de test pour les applications web (par exemple Selenium, utilisé très ponctuellement dans EDBA), d'outils permettant l'internationalisation (avec des outils type getText, utilisé pour la version anglaise d'EDBA) et l'aide à la documentation du code (avec code-illuminated) utilisé ponctuellement également.

Sur l'intégration dans les plateformes éducative : plusieurs difficultés sont à prendre en considération, les formats et technologies licites sur la plateforme, la forme des dépôts autorisés par la plateforme, etc. ainsi que les limitations liées à Javascript (par ex. la politique d'origine commune : le code Javascript d'une page ne peut accéder par

XmlHttpRequests qu'au serveur sur lequel se trouve la page). Pour faciliter les déploiements et permettre une intégration dans Moodle qui accepte les documents HTML, un outil supplémentaire a été mis en place pour EDBA permettant l'assemblage à la carte dans un unique fichier html de l'ensemble des fichiers html, js, css et même images nécessaires à l'application. Le développement peut donc ainsi se faire avec une multitude de petits fichiers et le déploiement avec tous ces fichiers ou en un seul fichier HTML selon un format accepté par Moodle. L'intégration dans Moodle nécessite également que l'accès à la BD soit possible. Au moins deux solutions existent, l'une remplace le type de BD habituellement utilisé, BD centrale, accessible via le réseau en général sur un serveur distinct de Moodle (ce qui n'est pas compatible avec la politique Javascript d'origine commune) par des BD locales à chaque intégration de l'application. Cette solution conserve les fonctionnalités de l'application mais produit des bases d'exercices divergentes. L'autre solution consiste à prendre une photo de la BD au moment de l'intégration, sous la forme d'un cache de requêtes, par exemple. Dans cette solution, les accès réseaux sont inactivés ce qui interdit les logs, le suivi de l'apprenant et l'évolution de la BD. Pour EDBA, c'est cette seconde solution qui a été choisie, l'une intégration dans Moodle se fait en un seul fichier (un fichier d'un peu plus d'1 Mo, avec 4 parties de tailles à peu prés équivalentes pour : les codes propres, les codes tirés d'emprunts, la base de données mise en cache, et les images).

L'association entre plateformes pédagogiques et EIAH est un point délicat qu'il faudrait approfondir. Sur les points abordés et sur d'autres encore. Quand l'EIAH, comme EDBA, ne comprend pas ou ne souhaite pas comporter de contenus conséquents (type éléments de cours ou exemples commentées à enseigner), l'apport des plateformes est évident. Ce cas se trouve pour la plupart des simulations, des micromondes, ou des exerciseurs plus portés sur l'activité. La plateforme pédagogique peut servir alors de lien entre contenus et activités.

2.2. Quelques choix, parfois discutables …

D'autres architectures sont possibles (PHP, Java) : au vue des considérations sur la transparence, les échanges qu'elle permet, l'intégration à Moodle qu'elle autorise au final, l'architecture choisie, avec l'essentiel de la partie applicative en Javascript, est préférable à d'autres formes d'applications AJAX basées sur PHP, ou sur des développements Java. Mais derrière ce choix se trouve également un enjeu pour le développement des EIAH : la volonté de réduire la complexité de développement des applications web en limitant le nombre de langages et d'environnements de développement nécessaires. Ainsi, le développement peut se faire en ne connaissant que Javascript, et quelques bases de SQL, HTML et CSS (seulement !) ; ces parties étant par ailleurs celles que l'on rend visibles dans les pages diffusées : l'utilisateur travaille donc avec les mêmes programmes que le développeur. L'interrogation d'une BD directement à partir de Javascript étant difficile (pour des contraintes relatives à la politique d'origine commune), une interface PHP est cependant probablement toujours plus ou moins nécessaire, mais cette partie PHP peut tenir en quelques lignes génériques, indépendantes de l'application. Ces lignes peuvent assurer de plus quelques fonctions de contrôle pour mettre en place un début de gestion de la sécurité, et le lien avec le courriel.

D'autres politiques vis-à-vis des bibliothèques Javascript : l'utilisation de Javascript, indépendamment du fait qu'elle peut tirer profit des programmes accessibles sur internet

(éditeur de code, terminal, interpréteur, …), est souvent associée à des bibliothèques (prototype, jquery, dojo, mootools, yui, ext, scriptaculous, …) Ces bibliothèques assurent habituellement une programmation plus simple et souvent plus efficace ; elles permettent de garantir une certaine indépendance vis-à-vis des navigateurs et des nouvelles versions de ceux-ci et sont parfois un indicateur de bonnes pratiques (mais « up to date » ou juste « à la mode » ?). Mais laquelle faut-il choisir ? et faut-il en choisir une ? Pour EDBA, il a été décidé de travailler sans bibliothèque. La qualité de ces bibliothèques n'est pas en cause. Cependant, les atouts offerts par ces bibliothèques sont à mettre en balance avec les constantes de vie de ces bibliothèques (ces bibliothèques disparaissent rarement, mais le nombre de mises à jour est important), la complexité qu'elles réintroduisent (chacune a sa façon de rajouter ses propres routines), la taille des codes que cela signifie (et parmi tout ce code, quelle partie sera vraiment utile ?). Pour un projet d'EIAH de longue haleine, qui ne souhaite pas avoir à surfer à la pointe de l'avant-garde technologique mais qui peut investir un peu sur les bases du langage qu'il va utiliser, le choix peut consister à réécrire les routines usuelles et vraiment utiles de ces bibliothèques en s'en inspirant éventuellement. La question de la compatibilité inter-navigateur est une question délicate, qui n'a pas été prise en compte dans le choix d'EDBA en arguant du fait qu'il s'agit d'un travail de recherche. Pour contourner ce problème, dans EDBA, une préférence explicite a été donnée à Firefox qui respecte assez bien les normes HTML/CSS/Javascript.

2.3. Quelques solutions encore à trouver et limites observées

Le réseau ? Diverses inquiétudes vis-à-vis de la performance existaient à l'origine du projet EDBA : le transfert des données sur le réseau, l'exécution de Javascript dans le navigateur seront-ils assez rapides ? Pour chaque application les réponses pourront différer, en ce qui concerne EDBA, pour une utilisation courante, ces inquiétudes n'avaient pas lieu d'être. Sans avoir particulièrement considéré ces questions, les premières versions d'EDBA étaient utilisables. Enhardi par ces résultats positifs, des utilisations gourmandes en données ou en temps de calcul ont été imaginées pour savoir jusqu'où l'on pourrait aller (par exemple pour des programmes ou traitements plus proche de l'administration de la BD que de l'utilisation de l'EIAH) : calculs divers pour l'ensemble des exercices enregistrés en base. La base d'exercices comportant plus de 200 exercices, cela signifie appliquer grossièrement un coefficient multiplicatif de 200 pour un traitement donné. L'objectif de ces essais consistait à chercher à atteindre les limites de ce qu'il serait possible de faire en quelques secondes au maximum dans une application AJAX. Selon les traitements, ces recherches de limites n'ont pas atteint leur objectif (la limite n'a pas été observée), ou ont effectivement montré l'existence de limite. Entre les performances du réseau et les performances de Javascript, le premier élément à céder, sur les quelques exemples pris, a toujours été le réseau. Cependant, un travail de réécriture des traitements en question, pour limiter les accès réseaux ont permis de repousser les limites à nouveau, par exemple l'accès à une centaine d'exercices, un par un, via une centaine de connexion donc, peut demander une trentaine de secondes, alors que l'accès au même ensemble via une seule connexion, se déroule en moins d'une seconde ; selon le traitement à effectuer ensuite, le gain en temps peut être suffisant pour faire disparaitre la limite. L'ajout d'un cache logiciel dans l'application peut aussi améliorer les performances.

La sécurité ? Quelques questions de sécurité/prévention ont été soulevées lors de la mise au point d'EDBA. Sécurité pour la BD distante, sur le serveur, soumise à d'éventuelles attaques d'utilisateurs malveillants. Assurance contre d'éventuelles tricheries d'étudiants, cf. [Rowe – 04], sur le poste client ou pour garantir une certaine confidentialité des données. Précautions contre des usages maladroits de l'application sur le poste client (ex : programmation par l'utilisateur d'une boucle infinie). Le langage Javascript a été conçu pour être relativement sûr vis-à-vis de l'extérieur. Non seulement, il ne doit pas permettre d'accéder à internet librement (seul le serveur d'origine de la page peut être interrogé), mais les autres onglets du navigateur, s'ils sont indépendants, sont également inatteignable ; idem pour le poste de l'utilisateur. Les applications Javascript s'exécutent dans un bac à sable. Cependant, pour AJAX, cet espace comporte aussi le serveur. Il faut donc protéger l'application contre l'utilisateur ; idem pour le serveur. Pour le serveur, une solution consiste à décider que les tables importantes ne soient pas accessibles en modification/suppression, que l'accès à ces tables ne soit possible qu'en lecture, et éventuellement, quand cela le nécessite, en ajout. D'autres solutions ponctuelles limitent le type de requêtes spécifiques autorisées. Pour éviter des tricheries, la meilleure solution est peut-être de ne pas avoir les solutions des exercices dans le logiciel. C'est le cas dans EDBA, des jeux d'essai avec des réponses de références sont données (elles sont mêmes explicites dans une définition étendue des problèmes accessible aux utilisateurs), mais les solutions (les algorithmes) n'existent pas explicitement pour l'utilisateur dans le programme. Un temps, il a été envisagé d'avoir une version cryptée d'une solution dans EDBA, avec la clé de cryptage externe à EDBA. Cette solution n'a pas été retenue, n'ayant finalement que peu d'intérêt pédagogique (il y a des livres ou des cours sur internet fournissant des solutions …) et contrariant l'indépendance d'EDBA vis-à-vis d'un langage de programmation spécifique (ces solutions auraient été dans un langage spécifique). Pour assurer un peu de confidentialité, un minimum d'informations personnelles peut être conservé dans l'application, et un mécanisme d'anonymisation doit être disponible pour effacer ses traces. Enfin, pour EDBA, afin de ne pas geler le navigateur avec un programme qui *boucle* (ce qui arrive souvent avec les débutants en algorithmique), les interpréteurs de langages disponibles ont été bridés pour n'effectuer qu'un certains nombres de pas de calcul. D'autres solutions sont à trouver ainsi, ponctuellement, dans chaque application, et il est clair qu'il y a des efforts à produire de ce coté là.

3. Expériences et perspectives pour EDBA et vis-à-vis d'AJAX pour EIAH ?

Commencé fin 2008, le projet EDBA a entamé ses premiers développements début 2009 ; depuis septembre 2009 quelques étudiants ont pu travailler avec les premières versions d'EDBA. L'année 2010 a eu pour objectif de passer des premiers prototypes d'EDBA, fonctionnant seulement pour Prolog en français sans administration à distance de la base d'exercices ni intégration possible dans Moodle, à l'application actuelle (multi-langue, multilingue, intégrable et avec administration à distance). Les utilisations concernent l'ensemble de la promotion de l'université Joseph Fourrier-Grenoble 1, L3-MIAGE-IMA de 2009-2010 et de 2010-2011 (en cours), soit environ 45 étudiants chaque année, ont eu un libre accès à EDBA. Seule une vingtaine a réellement saisi cette opportunité (+10 simples curieux n'ayant effectué qu'une visite rapide inférieure à 1/2h) avec des durées de travail

entre 1h30 et 15h, soit un total global d'un peu plus de 100h. À ce jour, un peu plus de 200 exercices sont disponibles dans EDBA avec près de 1500 jeux d'essai, soit environ 7 tests par exercice. Les étudiants ont laissé des logs comportant une dizaine de milliers de codes (codes intermédiaires, en cours d'édition, ou code plus définitifs, sujets à évaluation). Une séance de travail (avec une dizaine de volontaires), type TP, a également été organisée pour vérifier la bonne tenue de l'application lors d'utilisations simultanées. Cette séance s'est bien déroulée. La mise à jour de l'application a effectivement été d'une grande facilité, un grand nombre de versions ont été utilisées et diverses versions spécifiques ont pu être réalisées. Le développement s'est apparenté aux méthodes agiles avec intégration continue.

La mise au point des premières versions d'EDBA utilisables en l'espace de 6 mois conforte l'image d'AJAX et de Javascript comme plateforme possible pour le prototypage rapide d'applications. C'est cependant une expérience que nous avions déjà eu avec Aplusix lors de sa réécriture en 2000 (alors avec l'environnement de développement Delphi, reconnu alors pour permettre des prototypages rapides). Certains trouveront que 6 mois c'est long, d'autres que c'est raisonnable. L'existence de modules EIAH et d'architectures ou de squelettes d'EIAH comme EDBA permettra-t-il de réduire ce délai ? Une autre application AJAX, version QCM d'EDBA, a été produite. Elle partage une grande partie du code d'EDBA et démontre le coté générique de l'architecture d'application web adossée à une base d'exercices, peut-elle servir d'exemple pour le développement d'autres applications sur le même canevas ? Les choix des technologies et des architectures étant souvent contraints et affaires de goût, c'est peu probable, et elle n'a pas été écrite dans ce sens, cependant il serait intéressant, probablement, de développer de tels EIAH (QCM) pour servir d'exemple de développement d'EIAH (simples mais génériques) comme vitrines et modèles de telle ou telle technologie.

Références (page web accédée en Nov. 2010)

[Bouhineau – 97] Denis Bouhineau. *Construction automatique de figures géométriques et programmation logique avec contraintes*, Thèse Univ. Joseph-Fourier - Grenoble I. 1997.

[Bouhineau – 03] Bouhineau D., Bronner A., Chaachoua H., Huguet T. Analyse didactique de protocoles obtenus dans un EIAH en algèbre. Actes de la conférence EIAH 2003.

[Garrett – 05] Jesse James Garrett, «*Ajax: A New Approach to Web Applications.*», Adaptive Path LLC, 18. February 2005.

[Graf & List – 05] Graf, S. and List, B. *An evaluation of open source e-learning platforms stressing adaptation issues*. Fifth IEEE International Conference on Advanced Learning Technologies, ICALT 2005.

[O'Reilly – 05] O'Reilly, Tim. What Is Web 2.0. Design Patterns and Business Models for the Next Generation of Software. Web 2.0 Conference 2005.

[TIOBE] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[Rebai et al. – 07] Issam Rebai, Nicolas Maisonneuve, Jean-Marc Labat. *Un entrepôt pour stocker et rechercher des composants logiciels utiles aux EIAH.* Numéro spécial : Conceptions et usages des plates-formes de formation. STICEF Vol 12, 2005.

[Rowe – 04] Rowe, N.C. *Cheating in online student assessment: Beyond plagiarism*. Online Journal of Distance Learning Administration, N°7 (2), 2004.

[Trgalova et al. – 09] Jana Trgalova, Denis Bouhineau, Jean-François Nicaud, *An Analysis of Interactive Learning Environments for Arithmetic and Algebra Through an Integrative Perspective*, Int^{al} J^{al} of Computers for Mathematical Learning 14(3), Springer, 2009.