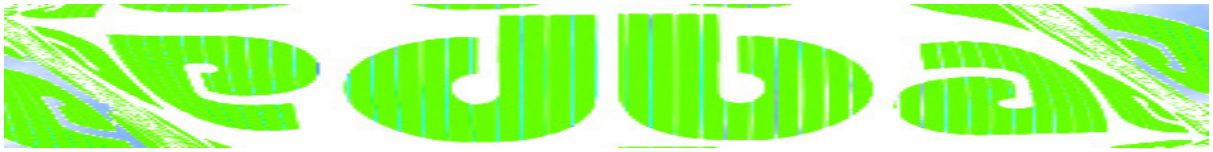


Projet pour une base d'exercices liée à un environnement d'apprentissage pour favoriser des développements mutuels via une démarche participative

Projet EDBA
-**E**xercises **D**ata**B**ase about **A**lgorithms-

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

Résumé. Prenant pour exemple les sites web d'exercices dans le domaine des mathématiques, le projet EDBA a pour objectif, dans le domaine de l'enseignement de l'informatique, de mettre en place une base d'exercices d'algorithmique associée à un environnement de programmation afin de permettre l'apprentissage de l'algorithmique. Le projet EDBA prévoit l'adoption d'une démarche participative (Web 2.0) pour la constitution de la base d'exercices ainsi que l'utilisation des technologies émergentes associées au Web 2.0 (AJAX).
Le développement est prévu en 3 phases. Lors de la première phase, la définition d'un mini environnement de programmation est planifiée. Associée à un scénario pédagogique simple de résolution d'exercices d'algorithmique et à une modélisation élémentaire des exercices d'algorithmique, l'application obtenue pourra s'intégrer à des plateformes d'e-learning (type Moodle). La seconde phase prévoit l'enrichissement de la notion d'exercice avec un début de structuration de l'ensemble, pris en tant que tel, des exercices et l'introduction de deux classes d'utilisateurs : les enseignants et les étudiants, les premiers pouvant participer à l'enrichissement de la base d'exercices, les seconds utilisant cette base pour progresser dans leur maitrise de l'algorithmique. La troisième phase prévoit l'introduction des aspects participatifs en dissolvant les deux classes d'utilisateurs pour mener à une classe unique d'individus ayant des expertises reflétant leur connaissance de l'algorithmique et de l'organisation d'EDBA. Chaque expertise sera associée au terme de cette phase à des droits et devoirs vis-à-vis de la gestion collaborative d'EDBA.
Mots-clés : EIAH, AJAX, Web 2.0, Algorithmique, Exercice, Base de données, Ressources pédagogiques, Démarche participative, Gestion collaborative, E-Learning.

Abstract. Following the example of educational websites in mathematics, the EDBA project aims, in the field of computer education, to establish a database for algorithmic exercises associated with a programming environment design for the learning of algorithms. The EDBA project plans to adopt a participatory approach (Web 2.0) for the development of the database of exercises and the use of emerging technologies associated with Web 2.0 (AJAX).
The development is planned in 3 phases. During the first phase, the definition of a tiny programming environment is scheduled. Combined with a starting scenario plan for a resolution of exercise relying onto an elementary model of the exercises, the resulting application could be integrated with e-learning platforms (like Moodle). The second phase involves enriching the concept of exercise with a beginning of structure for the set of all exercises, taken as such, and the introduction of two classes of users: teachers and students,

the first may participate in the enrichment of the database, the latter using this database to improve their mastery of algorithms. The third phase envisages the introduction of participatory aspects by dissolving the two classes of users to complete a single class of individuals with varied skills reflecting their knowledge of algorithms and organization of EDBA. Each skill will be associated, at the end of this phase, with rights and duties regarding the collaborative management of EDBA.
Keywords: TEL, AJAX, Web 2.0, Algorithm, Exercise, Database, Pedagogical resources, Participatory approach, Collaborative management, E-Learning.

Le projet EDBA, à destination de la communauté éducative -enseignants et élèves- en informatique, a pour objectif de mettre en place une base d'exercices d'algorithmique indépendante de la langue (Français, Anglais, ...) et des langages de programmation (ProLog, Caml, JavaScript, Ada, ASM, C, ...) associée à un environnement de programmation offrant plusieurs langues et langages pour apprendre l'algorithmique, la pratiquer, l'enseigner, l'étudier dans ses aspects épistémologiques, didactiques, cognitifs, culturels, ... et permettant l'enrichissement de la base d'exercices elle-même, via des démarches participatives.

En mathématiques, de nombreux projets de bases d'exercices pour l'apprentissage et l'enseignement des mathématiques ont vu le jour ces dernières années avec des succès significatifs :

- sur internet, depuis la banalisation des accès au web,
- en France, notamment, où la communauté éducative est forte et structurée,
- faisant suite à une abondante littérature 'papier' pour la préparation des examens et des concours,
- souvent avec des approches participatives et/ou collaboratives,
- souvent avec une possibilité donnée aux utilisateurs apprenant de faire ces exercices et de voir leur réponse évaluée,
- parfois basés sur des plateformes web d'enseignement généralistes, le plus souvent associés à des plateformes ad'hoc,
- ...

Citons en particulier :

- MathEnPoche [Lobato & Hache – 07, MathEnPoche], diffusé par l'association Sésamath, avec plus de 1000 exercices,
- Wims [Xiao – 01, WIMS], à l'université de Nice, avec plus de 500 exercices ou familles d'exercices,
- LeActiveMath [Melis et al. – 01, ActiveMath], en Allemagne, avec plus de 500 exercices et autant de définitions, théorèmes, illustrations,
- Matexo [Matexo], pour l'enseignement supérieur, avec plus de 300 exercices et ressources diverses,
- et d'autres encore : Cable, AIM, STACK ([Sangwin – 04] en Angleterre)
- …

En informatique, pour l'enseignement de l'algorithmique en particulier, à ce jour, il n'y a pas d'équivalent. Des ressources sont cependant disponibles, mais avec moins d'ampleur ou peu de structuration ou d'organisation pédagogique, sémantique ou épistémologique et une orientation qui est rarement tournée vers l'enseignement :

- Sans orientation pédagogique, indiquons :
    - Wikipedia [Wikipedia], l'article « liste des algorithmes » comporte, selon les langues, une centaine d'algorithmes (en français), ou plus de 500 algorithmes (en anglais),

- Dictionary of Algorithms and Data Structures [DADS], du National Institute of Standard and Technology du gouvernement américain, comprend plus de 300 algorithmes,
- Sphere Online Judge [SPOJ] propose plus de 5000 problèmes d'algorithmique à résoudre, issus de concours d'algorithmique et de programmation,
- The Computer Language Benchmarks Game [ShootOut] fournit plus de 1000 programmes pour comparer les langages de programmation
- le Projet Euler [ProjectEuler] compte de l'ordre de 300 problèmes d'algorithmique à connotation mathématique,
- …

- Plus spécifiquement pour l'enseignement de l'algorithmique, ou l'apprentissage de la programmation avec, dans tous les cas, un langage de programmation particulier, mentionnons :
  - Ninety-Nine Prolog Problems [P99] avec une centaine de problèmes d'algorithmique et des solutions en Prolog,
  - Elm-Art [Elm-Art] avec une cinquantaine de problèmes d'algorithmique en Lisp et une interface interactive permettant de s'exercer et de voir ses solutions analysées,
  - et d'autres encore pour Perl, Python, Ruby, Scheme (par ordre alphabétique)
  - …

En France, il y a une dizaine d'années, il y a bien eu Spedago, soutenu par Specif [Specif] société des personnels enseignants et chercheurs en informatique, dont l'objectif était de constituer un centre de ressources pédagogiques pour l'informatique et qui aurait pu devenir le centre de ressources pédagogiques national en informatique ; mais aujourd'hui il est difficile d'en retrouver une trace active et encore plus d'y accéder [Spedago-lien inaccessible]. Le chapitre informatique de wikiversité peine à se développer [Wikipedia] et manque l'objectif de la diversité des points de vue et approches.

Sans chercher à apporter une réponse approfondie à la question, évoquons quelques pistes de réponse pouvant être les explications de cette différence, de la rareté de l'offre pédagogique en informatique comparée à celle visible en mathématiques ?

Sur la diversité des langues & cultures et la diversité des langages & paradigmes. Comme en mathématiques, se pose le problème de la diversité des langues et des cultures. Dans le contexte de la recherche, ce problème peut ne pas être important ; dans la vie réelle, il prend une autre ampleur, c'est un problème à prendre très au sérieux [Libbrecht - 08].

En informatique s'ajoute la diversité des langages et des paradigmes des programmations. En moins d'un siècle, plusieurs centaines de langages ont été définies (plus de 500 sont cités sur wikipédia [Wikipedia]), l'un s'appelle même Babel [Moreno-Navorro & Rodriguez-Artalejo – 92]. L'informatique est une science jeune qui n'a pas encore convergé, si un jour elle doit le faire comme l'ont fait les autres sciences, vers un formalisme d'expression commun.

Notons que ce découpage en langages et paradigmes de programmation est aussi fort que le découpage en langues et cultures et qu'il lui est orthogonal, ce qui provoque un résultat désastreux. L'effort à fournir pour offrir une ressource dans P langages et pour Q langues est multiplié par le produit P*Q si chaque configuration comporte des différences à prendre en compte.

Sur la communauté éducative en informatique. En France, et dans de nombreux pays, l'essentiel de l'enseignement de l'informatique, et de l'algorithmique en particulier, est effectué dans l'enseignement supérieur. Cela signifie, comparé aux mathématiques, moins d'élèves, moins d'enseignants, moins de ressources et moins de recherche sur le domaine.

Pour donner une idée de l'ordre de grandeur de la différence entre mathématiques et informatique, en 2007 le ministère français de l'enseignement supérieur dénombrait environ 3000 enseignants en informatique parmi ses personnels (810 professeurs des universités, 2329 maîtres de conférences et 112 enseignants du second degré) alors que le ministère de l'éducation nationale comptabilisait 47 211 enseignants de mathématiques. Le rapport est de plus de 1 à 10.

À cela s'ajoute l'idée -à vérifier, à réfuter et éventuellement à combattre- qu'il y a moins de besoins pédagogiques dans l'enseignement supérieur, quelle que soit la discipline concernée. Ceci accentue la rareté des moyens et travaux disponibles pour la recherche en didactique et en pédagogie pour l'informatique et mène à ranger l'informatique parmi les disciplines orphelines pour l'accompagnement didactique et pédagogique des enseignements.

Au niveau international, la situation n'est guère plus florissante, [Pears & Al. 07] énonce que si la recherche en pédagogie pour l'apprentissage de la programmation existe, elle concerne le plus souvent des études locales et restreintes. Il ajoute de plus que cette recherche se diffuse mal et n'arrive pas jusqu'aux enseignants. Aussi, [Pears & Al. 07] pointe-t-il un besoin crucial d'études à grande échelle qui soient mieux diffusées et atteignent les enseignants en informatique.

Sur la recherche en didactique de l'informatique et l'épanouissement du web. Une dernière piste d'explication à explorer consiste à comparer l'évolution de la recherche en didactique de l'informatique et l'épanouissement du web.

L'enseignement de l'informatique en France n'a pas toujours été dans cette situation. Il fut un temps, suite aux plans « Informatique » du gouvernement français, où la recherche en didactique de l'informatique et l'encadrement des enseignements en informatique étaient importants. Avec [Baron & Bruillard – 01], nous situerions l'apogée de cette période en France dans les années 1990 avec, par exemple, les 3$^{\text{ièmes}}$ rencontres francophones de didactique de l'informatique [Baron et al. – 92].

À cette époque, dans les années 1990, le web était balbutiant mais il allait vite prendre son essor. Il n'était pas question, alors, de démarches participatives ou collaboratives sur internet. L'apparition du web 2.0 et son succès, quant à lui, est à situer au milieu des années 2000 [O'Reilly – 05], alors que la recherche en didactique de l'informatique et l'accompagnement de l'enseignement en informatique avait très largement entamé leur déclin.

De là à conclure que la didactique de l'informatique et le web participatif se sont croisés à quelques années de décalage sans pouvoir se rencontrer, … alors que la communauté mathématiques a su prendre en marche ce train qui partait.

Ajoutons, pour clore sur une note sémantique -malheureusement peu réjouissante pour ce qui nous intéresse- qu'aujourd'hui en France, le terme « didactique de l'informatique » signifie de moins en moins didactique de l'algorithmique ou de la programmation et de plus en plus didactique de l'utilisation des logiciels informatiques dans le domaine de l'éducation, comme si les mots même, en France, signifiaient la mort de la recherche en didactique de l'informatique pour ce qui concerne l'algorithmique. Ce phénomène n'est pas nouveau, il est visible depuis [Laborde - 88], mais il s'est accru ces dernières années.

Le projet EDBA a donc pour objectif de constituer une première base d'exercices d'algorithmique sur Internet destinée à la pratique de l'algorithmique, son apprentissage, son enseignement. L'exemple des sites de mathématiques semble prouver que l'élaboration d'une telle base est possible. Au delà des problèmes de langues et de cultures, reste le problème de la multiplicité des langages et paradigmes de programmation.

Pour mener à bien ce projet et se ramener à la situation des sites de mathématiques, l'un des efforts à produire consiste donc à surmonter le contexte difficile introduit par la multiplicité des langages informatiques, ou à trouver un niveau d'abstraction suffisant qui permette d'ignorer ces différences entre langages informatiques. En adoptant une démarche proche de celle employée dans les sites de concours d'algorithmique sur internet (cf. [SPOJ]) et courante dans les environnements de validation automatique de programmes [Pears & Al. - 07], il semble que cela soit possible. Cela peut se faire, par exemple, si l'on réduit un algorithme ou un programme à l'observation de ses entrées/sorties, c'est-à-dire si l'on assimile, en première approximation, le programme d'un utilisateur et l'ensemble des couples (questions, réponses) produit par ce programme sur un ensemble de tests ou jeux d'essai choisis *à la main* convenablement. Même si « Program testing can be used to show the presence of bugs, but never to show their absence! » dixit [Dijkstra – 69], on peut ainsi évaluer, en première approximation toujours, un programme d'élève écrit en C avec une solution de référence écrite dans un autre langage, par exemple en Pascal ou Java. On peut même évaluer une proposition de solution en l'absence de solution de référence programmée, dans la mesure où l'on connait les réponses ponctuelles aux tests que l'on pose.

Restent probablement, également, à résoudre quelques problèmes techniques spécifiques à l'informatique (comme il en existe en mathématiques), à mettre en place un site collaboratif pratique et attrayant (comme ont pu le faire les sites mathématiques), à remplir la base de quelques centaines d'exercices (mais quand ceux qui font la coquille sont ceux qui la remplissent, cela peut sembler plus facile, comme semble le montrer l'exemple des mathématiques) et trouver quelques milliers d'utilisateurs pour voir comment l'expérience peut vivre en grand.

La suite de cet article décrit comment le développement du projet EDBA peut se dérouler en plusieurs étapes à travers la réalisation successive de trois applications web :

- EDBA 0.0 : un premier environnement de programmation permettant de tester un programme vis-à-vis de jeux de tests enregistrés ;
- EDBA 1.0 : prolongeant ce qui précède avec une structuration de la notion d'exercice, l'enregistrement de ces exercices dans une base de données et l'introduction de deux classes d'utilisateurs : les enseignants et les étudiants, les premiers pouvant participer à l'enrichissement de la base d'exercices, les seconds n'étant que simples utilisateurs ;
- EDBA 2.0 : repartant d'EDBA 0.0 et de la base d'exercices obtenue avec EDBA 1.0 et introduisant des aspects participatifs et collaboratifs pour la gestion de la base d'exercices en supprimant la distinction entre utilisateurs-enseignants et utilisateurs-étudiants.

L'écriture de la première version de cet article s'est déroulée au début de l'élaboration de ce projet, afin d'en avoir une vision globale, de permettre des discussions internes, d'avoir un planning prévisionnel, même approximatif, et d'en évaluer l'intérêt et la réalisabilité. Ceci explique en partie le plan décrit précédemment, la forme parfois peu littéraire, le contenu très orienté vers les spécifications des différentes étapes du projet, un peu à la façon d'un cahier des charges ou des demandes d'un hypothétique client. Par ailleurs, ce contexte d'écriture – période préparatoire à la réalisation d'un projet- permet de comprendre et de relativiser le ton parfois polémique ou euphorique des propos et les objectifs qui peuvent sembler trop ambitieux.

Comme le lecteur pourra s'en rendre compte dans les pages qui viennent avec les sections sur EDBA 1.0 et EDBA 2.0, si l'informatique et l'algorithmique –comme matières enseignées- sont centrales dans le projet EDBA, à un niveau d'abstraction supérieur, ce n'est que la

discipline enseignée. Une grande partie de cet article pourrait être transposée à d'autres matières ou d'autres apprentissages, les mathématiques, l'orthographe, les QCM, …

Hors informatique, ce qui caractérise le projet EDBA est la recherche d'une association entre une base d'exercices et un environnement d'apprentissage permettant des enrichissements croisés. La conception de la base d'exercices, de l'environnement d'apprentissage et la mise en place des démarches participatives se veulent fondées sur des règles assez simples en grande partie indépendantes de l'informatique mais dépassant ce que l'on trouve habituellement sur les plateformes généralistes d'e-learning (type [Moodle]) qui s'apparentent –de ce point de vue- à de simples dépôts de documents pédagogiques sans scénarii pédagogiques d'apprentissage. Le projet EDBA est à mi-chemin entre ces dépôts simples et les environnements d'apprentissages pilotés par des scénarii, à la recherche de moyens pour suivre, susciter et motiver des apprentissages et tirer bénéfice d'une démarche participative visant enrichir les contenus et l'environnement lui-même.

Avertissement, rappel, mise au point : bien que rédigé la plupart du temps au présent de l'indicatif, excepté le prototype évoqué dans la partie EDBA 0.0, au moment de la rédaction de cet article, les différentes étapes du développement prévu ne sont pas encore réalisées et constituent les éléments d'un travail prospectif à mettre en place ; ce sont des préconisations pour un environnement d'apprentissage de l'algorithmique, des propositions d'applications imaginaires, des projets exploratoires, des projections d'un possible qui cherchent à être cohérentes et utiles mais qui restent des artefacts à construire.

EDBA 0.0 - Le cœur projet, sa base, est constitué par un mini-environnement de programmation destiné à la résolution d'un exercice d'algorithmique par un étudiant : EDBA 0.0 ou CheckEdit (ou TestEdit). Cet éditeur amélioré doit permettre à l'étudiant confronté à un exercice de programmation non seulement d'éditer son programme mais aussi de le tester vis-à-vis de jeux de tests enregistrés. Dans EDBA 0.0, l'exercice doit être identifié par le nom de la fonction ou du programme qu'il faut définir ; il est caractérisé par un énoncé en langage naturel, une spécification semi-formelle de l'entête de la fonction et des jeux de tests. Exemple d'exercice : recherche d'un élément dans une liste triée.

La forme générale du scénario de résolution d'un exercice imaginée pour EDBA 0.0 correspond à la démarche suivante en 7 étapes :

- première étape : prise de connaissance par l'étudiant de l'énoncé de l'exercice, cf. figure 4. Cet énoncé propose la recherche d'une « spécification, de la réalisation et de la définition de jeux d'essai » pour un problème d'algorithmique donné.
- seconde étape : après rédaction informelle par l'étudiant d'une spécification -par exemple dans l'éditeur de texte proposé dans EDBA 0.0 comme commentaire en début de programme donnant le profil de la fonction solution du problème- l'étudiant doit confronter ses choix de nom pour la fonction, pour le nombre de paramètres, leur place et leur type à la spécification donnée dans l'exercice. En cas de désaccord, dans la mesure où ces choix sont compatibles avec ceux d'EDBA, il faut adopter les choix proposés par EDBA. En cas d'incompatibilité, les retours sur la validité d'une solution fournis par EDBA ne pourront pas être produits. Cette étape, assez formelle et contraignante, est importante dans la mesure où elle spécifie le format du programme attendu, les paramètres, etc. EDBA repose sur ces informations pour valider une solution à l'aide de tests automatiques, prévus à une étape ultérieure, par comparaison entre l'exécution du programme proposé par l'utilisateur et une solution enregistrée. Eventuellement, l'ordre des paramètres pourraient être laissé flou, en imaginant une validation, modulo l'ordre des paramètres. Idem pour le choix du nom de la fonction,

qui peut être abstrait. L'introduction ou la suppression d'un paramètre dans la solution donnée par l'utilisateur par rapport à la solution enregistrée dans EDBA introduit une difficulté bien plus grande pour pouvoir comparer les deux programmes qui, dés lors, n'ont pas un profil commun, ni compatible modulo l'ordre des paramètres et le choix du nom de la fonction.

- troisième étape : description de jeux d'essai par l'étudiant et confrontation avec les jeux d'essai donnés dans l'exercice et inclus dans la base. C'est une étape innovante d'un point de vue pédagogique car elle introduit la notion de jeux d'essais à un niveau de la pratique pédagogique plus important que celui qu'on lui attribue habituellement. Cette étape, importante, consacre la place prise par les jeux d'essais dans EDBA (ce sont eux qui permettent la validation des propositions de solution)
- quatrième étape : programmation par l'étudiant d'une solution candidate au problème dans l'éditeur d'EDBA, exemple figure 1.
- cinquième étape : vérification libre par l'étudiant de la qualité de sa solution candidate par test sur ses jeux d'essai, les jeux d'essai enregistrés dans EDBA ou tout autre jeu d'essai et mise au point du programme candidat. Le tout se fait à l'aide de la console (i.e. terminal virtuel) proposée dans EDBA 0.0.
- sixième étape : test automatique par EDBA 0.0 de la solution candidate fournie par exécution automatique du programme candidat sur les jeux d'essai donnés dans l'exercice et comparaison automatique avec les solutions enregistrées attendues, cf. figure 2. Prise de connaissance du résultat par l'étudiant. Si l'étape 5 a été bien suivie, le résultat doit toujours être un succès complet. L'objectif est d'insister sur l'importance d'un travail de validation de l'algorithme avant de rendre sa copie.
- septième étape : confrontation de la solution candidate de l'étudiant avec la solution fournie par le système (quand celle-ci existe), cf. figure 5.

À ces sept étapes, il faut ajouter une étape initiale pour le choix de l'exercice. Celle-ci peut correspondre à la demande d'un enseignant, à une étape d'un scénario d'apprentissage d'une plateforme d'enseignement, à un choix libre de l'étudiant dans la liste des exercices disponibles (cf. figure 4) ou à toute autre modalité.

Les éléments nécessaires à la bonne marche de ce scénario sont donc :

- la présence d'un exercice et des éléments associés (énoncé, spécification, jeux d'essai et, de manière optionnelle, solution) ;
- l'existence d'un éditeur de code ;
- l'existence d'un terminal ;
- l'existence d'un compilateur ou d'un interpréteur associé à l'éditeur et au terminal ;
- l'existence d'un comparateur lié aux éléments précédant permettant de valider la production de l'étudiant.

Les défis techniques de cette étape consistent à fournir EDBA 0.0 sous la forme d'une page web unique, à base de technologies Ajax, facilement intégrable à une plateforme d'enseignement web, type Moodle [Moodle]. L'existence, en javascript, d'interpréteurs d'une dizaine de langages informatiques ainsi que d'éditeurs de code et de terminaux, également écrits en javascript, permettent de penser que cet objectif est réalisable sous la forme d'un ensemble de pages web. Il manque seulement le liant entre ces différents éléments et un comparateur. Tout ceci peut se réaliser en Javascript, l'assembleur des applications web. La nécessité de rassembler l'ensemble de ces pages en une page unique pour pouvoir l'intégrer plus facilement à une plateforme d'enseignement complique cependant significativement le problème.

Pour une intégration plus complète dans les plateformes d'enseignement, EDBA 0.0, modélisé comme un micromonde de programmation, doit pouvoir interagir avec la

plateforme ; avec, venant de la plateforme, les informations nécessaires à EDBA 0.0 pour savoir quel exercice proposer à un élève (étape initiale du scénario de résolution d'un exercice comme présenté précédemment) et dans le sens inverse, en direction de la plateforme, des indications d'EDBA 0.0 pour informer la plateforme de l'activité de l'étudiant, suivre cette activité, les résultats aux exercices et sauvegarder son état courant.

Le mini-environnement de programmation EDBA 0.0 doit donc -pour un exercice donné et un étudiant spécifique- comporter ou permettre l'accès à :

- l'énoncé informel de l'exercice dans la langue naturelle de l'étudiant, avec un ou plusieurs exemples, également informels, des questions et solutions attendues,
- une spécification plus formelle de cet énoncé (entête de fonction, type des données, des résultats, ...),
- des jeux d'essai ou exemples formels d'exécution d'un programme solution de l'exercice, choisis *à la main* par le concepteur de l'exercice, avec des attributs précisant l'importance quantitative et qualitative de chaque test, les résultats attendus et -pour certaines erreurs courantes prévisibles- des résultats incorrects prévisibles accompagnés de diagnostics d'erreur,
- [de façon optionnelle] une solution, cryptée, rédigée dans un langage informatique spécifique (choisi par l'étudiant parmi ceux disponibles),
- divers outils pour travailler l'exercice :
    - un éditeur de code connaissant la syntaxe du langage informatique utilisé, permettant la coloration automatique, la complétion automatique, des exemples de code, de la documentation, …
    - une console ou terminal possédant un interpréteur et un debugger du langage informatique utilisé, permettant à l'étudiant de tester son programme et d'effectuer les tests unitaires associés à l'exercice (en cas de différence significative entre la solution de l'étudiant et la solution donnée par le programme solution de référence, un message doit être émis et la solution 'de référence' doit être donnée)
    - une rétroaction permanente non intrusive réduite aux résultats ok/~/ko des tests disponibles au niveau de l'éditeur (par exemple avec un feu tricolore –sur des zones de code- rouge : à peu près sûr qu'il y a une erreur, orange : diagnostic pas clair, vert : à peu près sûr que c'est correct)

Un prototype d'EDBA 0.0 est disponible pour la pratique de certains exercices en Prolog en français (en cours de construction) :
http://www.noe-kaleidoscope.org/public/people/DenisB/Enseignement/Prolog/pl_3in1.html
avec :

- un prototype d'éditeur (cf. figure 1),
- un interpréteur pour Prolog,
- un prototype de console (cf. figure 2),
- une fonction de test automatique (cf. figure 2) et
- un lien vers une liste d'exercices stockés dans un dépôt de fichiers web généraliste sous forme de fichiers html (cf. figure 3) comportant :
    - un énoncé informel (cf. figure 4),
    - une spécification formelle du programme recherché (cf. figure 5),
    - des tests unitaires (cf. figure 5),
    - une solution (cf. figure 5).

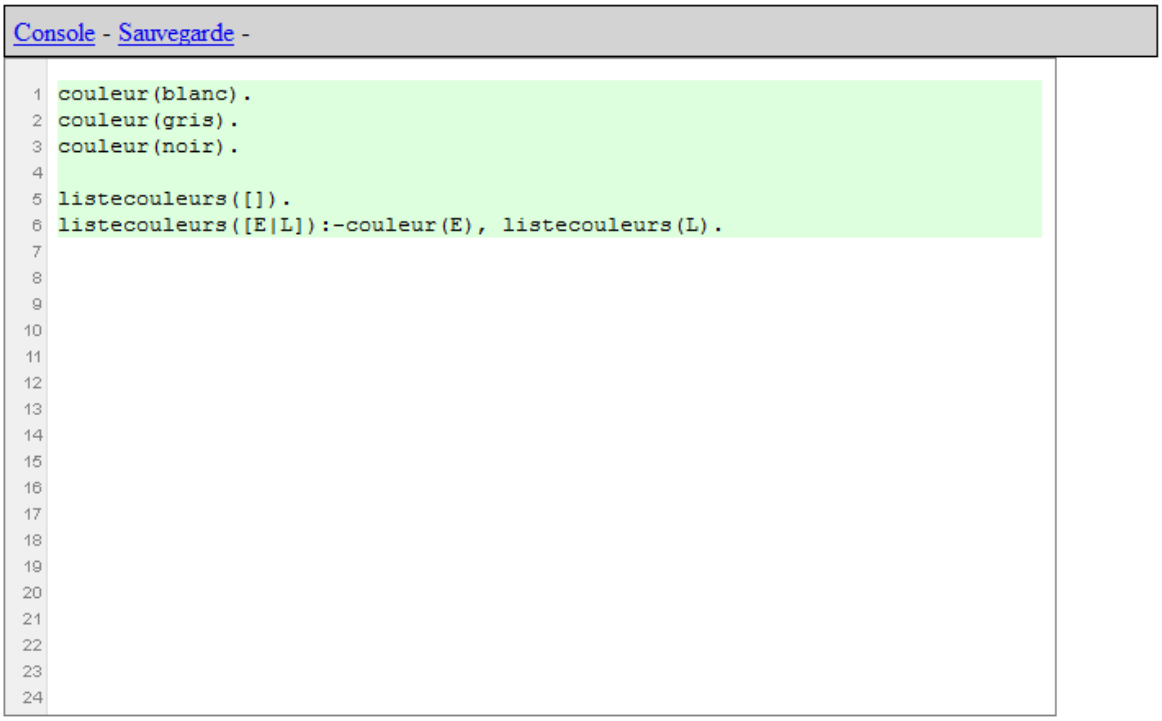
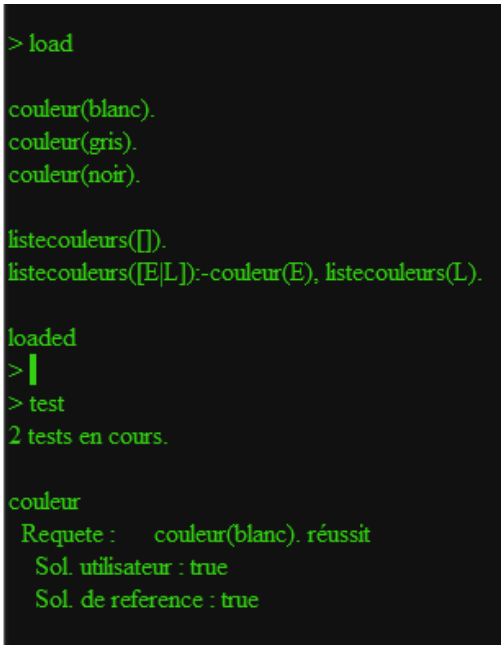


Figure 1 : L'éditeur prototype pour EDBA-0.0



[…]

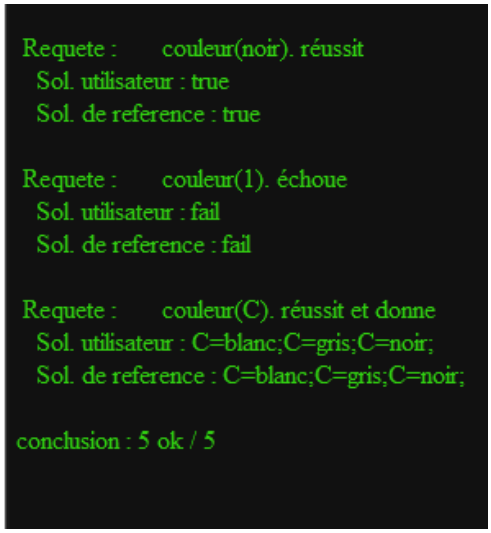


Figure 2 : La console prototype pour EDBA-0 avec un test automatique

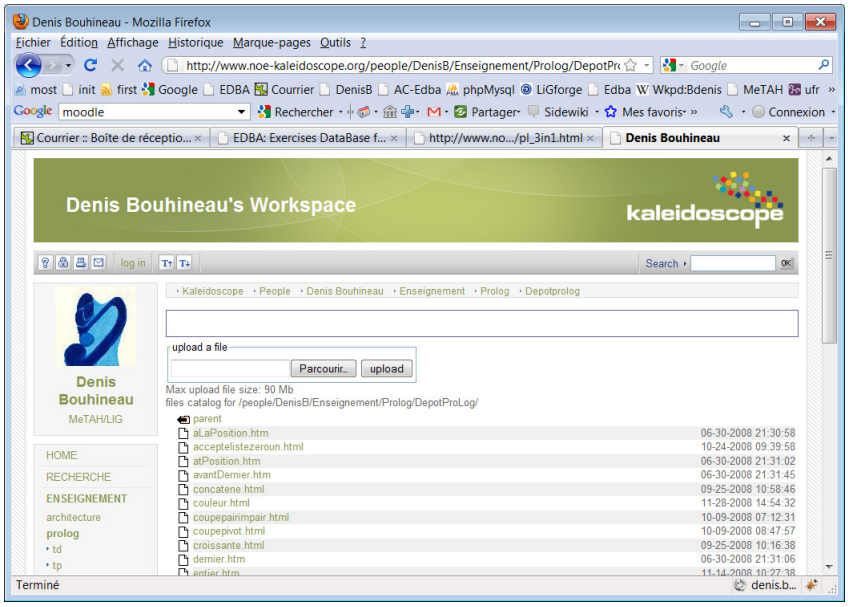


Figure 3 : dépôt de fichier (outil standard d'un CMS)

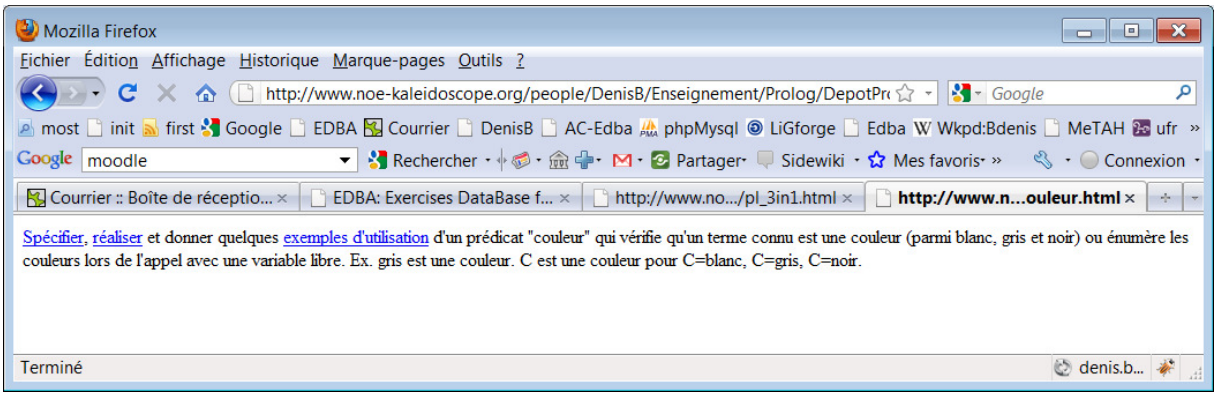


Figure 4 : fichier HTML prototype pour sauvegarder un exercice
(une partie du contenu est caché, pour y accéder,
l'utilisateur doit cliquer sur les pseudo-liens en bleu, cf Figure 5.)

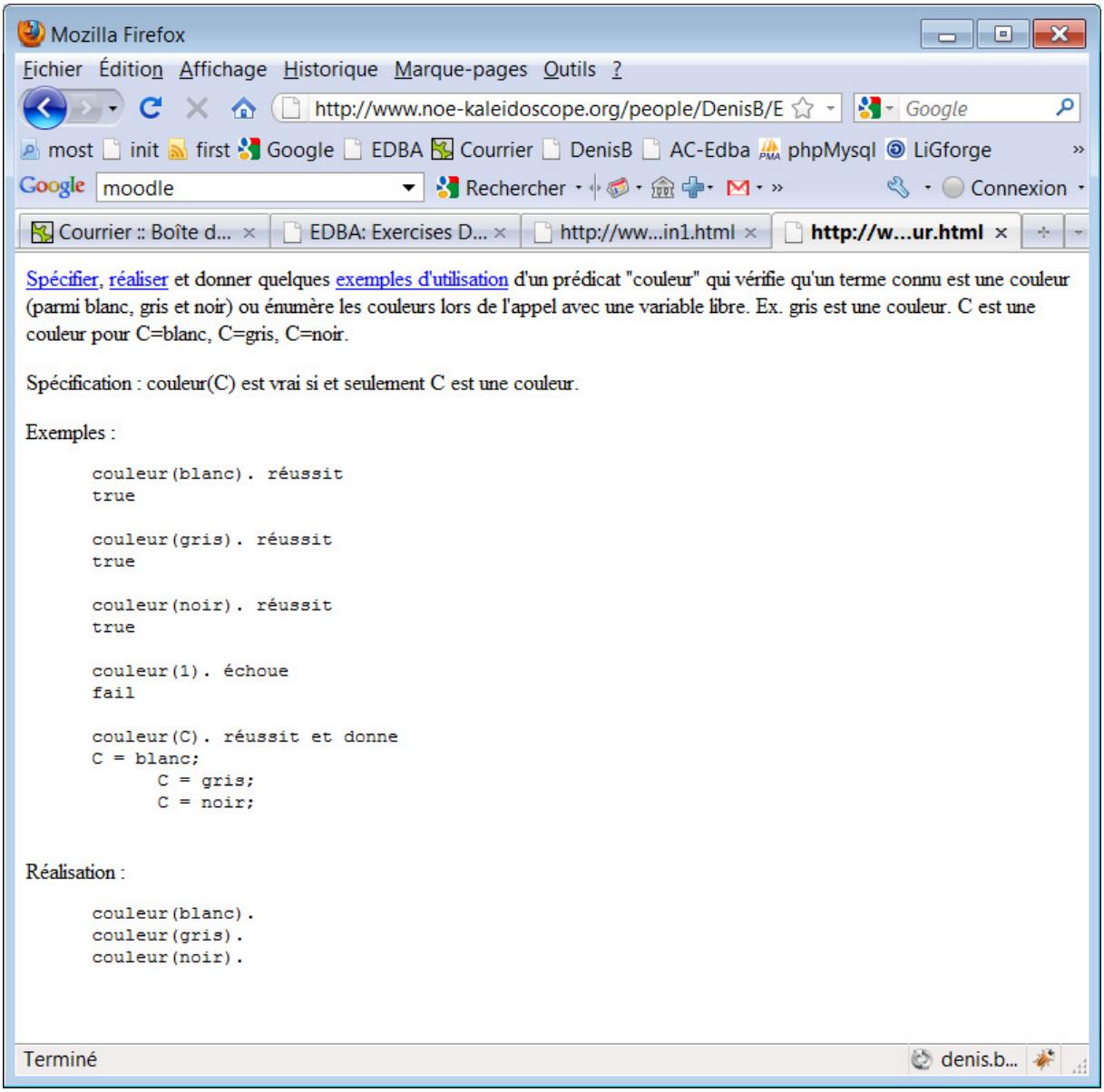


Figure 5 : fichier HTML prototype pour sauvegarder un exercice
avec affichage du contenu caché

Aucune étude sur l'utilisation de ce prototype n'a été effectuée. La réalisation de ce prototype n'avait pour autre objectif que d'assurer la faisabilité du projet EDBA. Il n'était pas destiné à expérimentation.

EDBA 1.0 –Niveau de maîtrise algorithmique des apprenants et niveau de difficulté algorithmique des exercices.

Un premier niveau d'organisation de la base des exercices du projet EDBA et des utilisateurs de cette base est donné avec EDBA 1.0.

Du point de vue technique, cela signifie la conception d'une base de données pour les exercices, les utilisateurs et, à la croisée des exercices et des utilisateurs pour des données relatives aux traces d'activité. À cela s'ajoute qu'EDBA 1.0 doit être capable de gérer des communications réseau et de dialoguer avec une base de données en lecture et en écriture.

EDBA 1.0 – Structuration du coté des utilisateurs

Chaque utilisateur doit pouvoir être identifié dans EDBA 1.0 par un login et un mot de passe. Eventuellement, l'identification doit pouvoir être déportée sur un site d'identification, d'affichage et de constitution de profil, par exemple selon le modèle OpenId [OpenId]. Au login est associé un pseudo, un nom réel, une adresse de courriel vérifiée à l'inscription et un profil public.

Deux classes statiques distinctes d'utilisateurs sont prévues : les enseignants et les étudiants. Les enseignants ont droit de lecture/écriture sur les exercices de la base. L'enrichissement de la base et l'organisation des contenus sont laissés à l'expertise des enseignants. Les étudiants n'ont qu'un droit d'accès et de lecture de la base d'exercice. Ce droit est, de plus, limité aux exercices correspondant à leur maîtrise algorithmique.

La séparation des utilisateurs en deux classes est une esquisse réduite du modèle final de l'utilisateur d'EDBA pour isoler et simplifier les fonctionnalités de modification et d'enrichissement de la base d'exercices. Dans EDBA 1.0, ces fonctionnalités sont réservées aux enseignants, c'est-à-dire aux experts et l'on suppose qu'il n'y a pas de conflits entre experts. Le niveau de développement suivant, EDBA 2.0, reviendra sur ces différents points - la séparation, la simplification, l'absence de conflit- pour introduire les éléments nécessaires à une démarche participative et à un travail collaboratif.

Pour l'étudiant, l'introduction de la base d'exercices modifie de manière transparente l'accès aux exercices. Le scénario d'usage est donc le même.

L'ajout, par les enseignants, de nouveaux exercices dans la base d'exercices peut se faire, soit ex nihilo à partir d'une page blanche informatique, soit à partir d'un exercice déjà présent dans EDBA par traduction, translation ou modification.

La transformation d'un exercice d'EDBA peut être légère et ne pas changer la nature de l'exercice, sa sémantique. C'est le cas lors des traductions dans une autre langue (ex. : français→english) d'un énoncé et éventuellement de la spécification d'un exercice et lors des translations vers un autre langage (ex. : Prolog→Caml), c'est-à-dire lors des changements de formulation et de vocabulaire dans un énoncé pour employer le jargon, les formes et formules correspondant à un langage ou un paradigme de programmation particulier. Dans ces deux cas, il n'y a pas, à proprement parler, de création d'un nouvel exercice, mais juste l'ajout de nouvelles formulations pour le même exercice. Les jeux d'essai, la difficulté de l'exercice, les concepts associés ne sont pas modifiés, ni dupliqués.

Les modifications peuvent être plus profondes, comme par exemple l'ajout d'une contrainte d'ordre sur les données ou sur les résultats. Alors, les jeux d'essai et la difficulté de l'exercice doivent aussi être mis à jour. Il y a alors, véritablement, création d'un nouvel exercice. Un lien avec l'exercice de départ doit cependant être conservé à toute fin utile.

La création d'un exercice ex-nihilo doit aussi pouvoir être possible. Une interface spécifique doit pouvoir être fournie à l'expert pour que l'écriture de l'énoncé de l'exercice, de la spécification, d'une solution et des jeux de test soit facilitée. Elle peut être basée sur EDBA 0.0 avec, par exemple, une console enregistrant les tests effectués lors de la mise au point de la solution pour les proposer comme jeux de test de l'exercice.

Dans tous les cas, l'écriture ou la modification d'exercices dans EDBA doit être facile et intuitive. De plus, elle doit être rapide. Une dizaine de minutes doit être un maximum pour rédiger sous forme électronique un exercice *simple*, déjà prêt sur le papier ou dans la tête, avec sa spécification et ses jeux d'essai.

Ces nouveaux exercices ou ces nouvelles formulations d'exercices doivent être proposés aux gestionnaires du site avant publication pour être validés. La validation pour ces derniers peut être directe ou nécessiter une phase de discussion avec les gestionnaires du site.

Le profil des enseignants est enrichi par diverses informations publiques :

- un texte libre (descriptif) (éventuellement en HTML) [optionnel] ;
- le lien vers une page web (extérieur au site EDBA) [optionnel] ;
- la liste des contributions de l'enseignant

et des fonctionnalités (utilisables par l'enseignant seulement) :

- l'accès aux travaux anonymés des élèves sur les exercices introduits par l'enseignant ;
- la possibilité d'envoyer un message aux étudiants pratiquant ces exercices (tout en conservant l'anonymat de ces élèves) ;
- la possibilité de construire une fiche de TD/TP à partir d'une sélection d'exercices (fiche + correction).

Le profil des étudiants (email, …) doit être facilement extensible, il est enrichi par diverses informations publiques ou privées (en gras les éléments importants) :

- un texte libre (descriptif) (éventuellement en HTML) [optionnel]
- le lien vers une page web (extérieure à EDBA) [optionnel]
- la maîtrise algorithmique de l'étudiant, pour chaque langage informatique, comptée en points d'Xp de maîtrise algorithmique pour le langage considéré, calculée en fonction des exercices abordés par l'étudiant (voir plus loin)
- le niveau de maîtrise algorithmique, pour chaque langage informatique, donné sur une échelle de 1 à 100, calculé en fonction du nombre de points d'Xp de maîtrise algorithmique pour le langage considéré : ce niveau de maîtrise algorithmique pour un langage donné est un résumé du nombre de points d'Xp de maîtrise de ce langage sur une échelle facilement compréhensible (voir plus loin)
- la maîtrise algorithmique absolue de l'étudiant, comptée en **points d'Xp de maîtrise algorithmique absolue,** calculée en fonction des maîtrises algorithmiques acquises pour chaque langage informatique (de type somme ou moyenne pondérée ou maximum, mais avec probablement une plus grande importance donnée aux valeurs les plus grandes)
- **le niveau de maîtrise algorithmique absolue** donné sur une échelle de 1 à 100, résultant de la maîtrise algorithmique absolue, calculé en fonction des niveaux de maîtrise algorithmique acquis pour chaque langage informatique (de type somme ou moyenne pondérée ou maximum, mais avec probablement une plus grande importance donnée aux valeurs les plus grandes) : **ce niveau de maîtrise algorithmique absolue est le repère essentiel du niveau algorithmique d'un utilisateur**, il doit permettre de faire un lien avec le niveau de difficulté des exercices, également donné sur une échelle de 1 à 100 (aussi appelé en abrégé « niveau d'un exercice »).
- le niveau d'exercice maximum accessible par l'étudiant, calculé en fonction du niveau de maîtrise algorithmique absolu de l'étudiant avec comme valeur minimum : le niveau de l'exercice résolu par l'étudiant précédemment le plus élevé + 3 et comme valeur maximale : le niveau de l'exercice résolu par l'étudiant précédemment le plus élevé + 10. Ce sont les **règles d'accès aux exercices**. Elles auront une grande importance dans la suite.
- le nombre de point d'Xp de maîtrise algorithmique nécessaire, pour chaque langage, pour atteindre un niveau de maîtrise algorithmique supérieur (et donc par conséquent pour atteindre un niveau d'exercice supérieur)

L'étudiant doit pouvoir accéder à trois fonctionnalités :

- la sauvegarde des exercices travaillés (état courant, états significatifs -corrects syntaxiquement) et du temps de travail effectif passé sur chaque exercice (accessible par l'étudiant seulement) ;

- le décryptage et l'utilisation des solutions des exercices ayant un niveau inférieur au niveau d'exercice accessible -12 ;
- la levée partielle de son anonymat via EDBA pour donner son nom réel et son adresse de courriel à un enseignant d'EDBA.

EDBA 1.0 – Structuration des exercices

La notion d'exercice est enrichie par

- les références des auteurs de chaque partie (énoncé, spécification, tests, solution) ;
- une liste de mots clés librement choisis [optionnel] ;
- la référence à un espace de nom pour la définition des fonctions auxiliaires utilisées [optionnel] ;
- le classement de l'exercice sur une échelle de difficulté des exercices d'algorithmiques ayant une centaine de niveaux (voir par exemple, la proposition faite dans le tableau 1).

Niveaux de difficulté algorithmique des exercices ou problèmes	**Types des exercices ou des problèmes**
de 0 à 3	exercices sur les expressions et l'affectation
de 4 à 10	exercices simples sur les boucles et les conditionnelles (une seule structure de contrôle)
de 11 à 17	exercices avec 2 structures de contrôle différentes (ex. une conditionnelle dans une boucle) ou consécutives (ex. deux boucles, l'une après l'autre)
de 18 à 24	exercices avec 2 boucles imbriquées (et éventuellement des conditionnelles)
de 25 à 31	exercices nécessitant l'introduction de fonctions/procédures auxiliaires (pour éviter un trop grand nombre de boucles imbriquées, ou la répétition d'un code récurent)
de 32 à 37	exercices complexes demandant un effort de modélisation, l'introduction de fonctions/procédures auxiliaires, ...
de 38 à 44	problèmes simples, comportant plusieurs exercices simples (niveaux 15 ou moins)
de 45 à 51	problèmes non simples, comportant plusieurs exercices dont certains non simples (niveaux 16 à 30)
de 52 à 58	problèmes complexes demandant un effort de modélisation
de 59 à 70	problèmes complexes demandant un effort de modélisation et la gestion de la performance, d'un embryon d'IHM, ou d'IA
de 71 à 80	problèmes complexes (cf. préc.) avec bd ou réseau
de 81 à 90	problèmes complexes (cf. préc.) avec utilisation réelle
de 91 à 100	EDBA 0.0, ..., EDBA 3.0

Tableau 1 : Echelle des niveaux de difficulté algorithmique des exercices.

Remarque : Il serait tentant d'appeler cette échelle « échelle de complexité des exercices » ou même « échelle de complexité algorithmique » plutôt que « échelle de difficulté des exercices » (en bref) ou des « échelle des niveaux de difficulté algorithmique des exercices » (en long), mais la confusion avec la complexité algorithmique –au sens où on l'entend en algorithmique, c'est à dire mesure des ressources, temps-espaces, nécessaires à l'exécution

d'un algorithme jusqu'à son terme– serait trop grande. Il s'agit bien de la complexité au sens courant et usuel du terme : de la complexité ou difficulté ressentie pour tout individu pour résoudre un exercice ou un problème donné, d'une complexité humaine, ressentie subjectivement. Il ne s'agit pas de la complexité algorithmique au sens informatique du temps. La distance entre ces deux termes est importante, même s'il peut exister quelques liens entre ces deux notions, parfois.

Seconde remarque : l'évaluation de la difficulté d'un exercice, valeur subjective, est laissée dans un premier temps aux experts. Dans une phase ultérieure, l'évaluation de cette valeur pourra reposer sur l'étude des traces de résolution du dit exercice.

EDBA 1.0 – Lien entre exercices et utilisateurs : entre niveau de difficulté algorithmique des exercices et niveau de maîtrise algorithmique absolue d'un utilisateur.

Le calcul du niveau de maîtrise algorithmique d'un étudiant et l'attribution de points d'Xp de maîtrise algorithmique à un étudiant sont des éléments clés d'EDBA. Les développements qui vont suivre risquent de paraître un peu longs et fastidieux ; il y a de nombreux facteurs à prendre en compte et, même si le résultat est assez simple, y arriver demande un peu de temps.

Coté étudiant, les points d'Xp de maîtrise algorithmique doivent avoir une double fonction : d'une part, fournir des éléments d'auto-évaluation de l'apprentissage de l'étudiant, d'autre part, être une source de motivation pour l'étudiant. Coté EDBA, il s'agit de mettre en correspondance l'échelle des niveaux de difficulté algorithmique des exercices avec les niveaux de maîtrise algorithmique absolue d'un étudiant (ces derniers dépendent de l'attribution de points d'Xp de maîtrise algorithmique).

Pour le choix des mécanismes mis en œuvre et leurs liens avec les effets escomptés du coté de la motivation, la démarche est empruntée au domaine des jeux vidéo, de type jeu vidéo de rôles sur internet (en anglais : role playing game, « rpg », ou massively multiplayer online role-playing game, « mmorpg », voir [Wikipedia]). Elle consiste à emprunter le vocabulaire et les pratiques développés dans le domaine des jeux pour motiver le joueur à continuer de chercher à progresser, en même temps que de l'informer sur sa progression via des récompenses dans le jeu. Les jeux pris en exemple sont de type jeux de rôles ou jeux à niveaux avec avatar : « Le *joueur est représenté par un avatar, personnage qu'il crée puis fait progresser dans un monde virtuel d'inspiration fantastique, de science-fiction ou de super-héros, riche en aventures. Ce faisant, il interagit avec l'environnement contrôlé par le programme et avec les autres joueurs. […] Beaucoup de MMORPG reprennent le concept des points d'expériences, afin d'atteindre de nouveaux niveaux ou compétences permettant à son personnage de devenir meilleur en de nombreux points. Cette progression se fait de deux manières distinctes. Soit en accomplissant des quêtes, c'est-à-dire des missions données par un personnage non-joueur, soit en tuant des monstres gérés par l'environnement. Ces deux actions rapporteront des points d'expérience (XP), le plus souvent en quantité proportionnelle à la difficulté de la tâche.* » [Wikipedia]

Les caractéristiques des personnages des jeux de rôles correspondent, dans EDBA, aux niveaux de maîtrise algorithmique (et plus loin, aux Droits&Devoirs). L'attribution de points d'Xp de maîtrise algorithmique lors de la résolution d'exercices permet de progresser ; cela correspond aux quêtes et combats. Les niveaux de jeux sont les niveaux de difficulté algorithmique des exercices accessibles par l'apprenant.

Il s'agit de faire sentir à l'utilisateur étudiant que la pratique de l'algorithmique le fait progresser. Il faut qu'il y trouve aussi une motivation pour continuer à travailler ; comme dans le jeu, la course au niveau supérieur motive le joueur pour continuer à jouer.

Pris dans toute sa complexité, le calcul du niveau de maîtrise de l'étudiant en algorithmique peut se faire à partir de l'étude de la trace des exercices abordés par l'étudiant, en prenant en compte la difficulté algorithmique des exercices abordés, leurs particularités thématiques ou les enchainements observés, le nombre de tests validés et manqués, l'importance individuelle a priori des tests mis en jeu (certains tests peuvent être plus importants que d'autres) et la qualité des validations effectuées (certaines validations peuvent être plus convaincantes que d'autres).

L'essentiel de ce calcul peut cependant reposer uniquement sur la difficulté des exercices résolus et le nombre de tests validés.

Toutefois, l'obtention des points d'Xp de maîtrise algorithmique en fonction de la difficulté algorithmique d'un exercice et du nombre de tests validés n'a pas de raison d'être linéaire vis-à-vis de l'une ou l'autre de ces variables.

Par exemple, pour la dépendance vis-à-vis du nombre de tests validés, le gain peut dépendre de l'exercice : réussir 80 à 90% de tests d'un exercice peut signifier que la résolution de l'exercice est bien avancée sans être acquise à 80-90%, les 10 à 20% manquant sont peut-être tout à fait nécessaires et représenter les cas les plus généraux. De même, 30 à 40% de succès aux tests d'un exercice ne doit pas signifier que l'exercice est résolu à 30-40% ; dans certains cas -comme pour des QCM n'ayant que peu de choix proposés- cela peut représenter seulement l'effet du simple hasard. Par ailleurs, le nombre de points maximal que l'étudiant peut obtenir lors d'un test peut décroitre en fonction du nombre de tests déjà tentés. Le premier essai sera avec un potentiel de points maximum. Les suivants se feront avec un potentiel de points décroissant.

Pour la dépendance vis-à-vis de la difficulté de l'exercice, il faut fixer, même approximativement, le lien entre les niveaux de difficulté algorithmique d'un exercice et l'échelle des niveaux de maîtrise algorithmique d'un étudiant. Pourquoi ? L'idée de cette échelle des niveaux de difficulté algorithmique des exercices est de proposer une échelle globale et compréhensible de la difficulté pour la résolution d'un problème donné. Ce qui est proposé, est une échelle de difficulté des exercices à 100 niveaux (cf. table 1.). L'échelle des niveaux de maîtrise algorithmique comporte également 100 niveaux mais n'a pas de signification intrinsèque. Elle a pour but d'être un résumé des points d'Xp de maîtrise algorithmique acquis par les utilisateurs lors de la résolution d'exercices, une échelle globalisante, et facilement compréhensible de la maitrise algorithmique d'un utilisateur. Le plus simple consiste à identifier ces deux échelles de niveaux, ou tout au moins à les faire coïncider autant que possible de telle sorte que l'accès pour un étudiant à un niveau de maîtrise algorithmique N signifie plus ou moins la maîtrise des exercices de niveau de difficulté algorithmique N.

Il reste à définir les modalités effectives d'attribution des points d'Xp de maîtrise algorithmique et le lien entre ces points d'Xp de maîtrise algorithmique et les niveaux de maîtrise algorithmique.

Pour ce dernier point, nous avons choisi une échelle à 100 niveaux de maîtrise algorithmique subdivisée en d'autant plus de points d'Xp de maîtrise algorithmique que le niveau est élevé. Le choix de cette échelle sur-linéaire n'a pas de justifications fortes ou pédagogiques particulières autres que l'observation des pratiques employées dans le domaine des jeux et l'analyse des conséquences positives qu'elle entraine. Ainsi, parmi les conséquences de ce genre d'échelle, il y a l'impression d'une progression qui est de plus en plus importante au fur et à mesure de l'avancée dans l'application. La réussite, dans cette progression sera donc d'autant plus valorisante qu'elle représentera une étape plus importante. L'impression peut toutefois être trompeuse : il suffit de ramener ce qui précède à une échelle logarithmique pour obtenir des progressions linéaires.

Associée à cette échelle sur-linéaire, pour que la progression soit effectivement possible en des temps à peu près proportionnels à la progression (ce qui signifie que la progression réelle se rapproche en fait du linéaire), l'attribution effective de points d'Xp de maîtrise algorithmique pour la réalisation correcte d'un exercice doit être également sur-linéaire en fonction de la difficulté de l'exercice. Ce dernier point doit également donner une impression positive à l'étudiant, qui verra ses gains progresser au fur et à mesure de l'avancée dans l'application. Par ailleurs, ce type d'échelle pousse l'utilisateur vers les défis ayant un niveau de difficulté supérieur –puisque la fonction de gain des points d'Xp de maîtrise algorithmique est croissante en fonction de la difficulté– ce qui favorise à nouveau la progression. Cette forme d'attribution sur-linéaire peut se justifier d'un point de vue pédagogique dans la mesure où elle permet de distinguer significativement des exercices de complexité algorithmique différente.

Par exemple, l'attribution effective de points d'Xp de maîtrise algorithmique choisie pour la résolution d'un exercice pourrait être donnée par le carré du niveau de difficulté algorithmique de l'exercice, avec un facteur multiplicatif lié à la proportion de tests évalués positivement lors des essais. En prenant des niveaux de maîtrise algorithmique échelonnés selon le quadruple du carré de chaque niveau, cela donne pour atteindre un niveau de maîtrise algorithmique supérieur, la résolution d'environ 4 exercices de niveau de difficulté algorithmique équivalent au niveau de maîtrise algorithmique de l'étudiant, dans la mesure où chaque test d'un exercice est validé. Une autre fonction croissante (le cube, …) peut être choisie, un autre facteur (2, 3, …) également.

Avec les choix effectués dans le paragraphe précédent, les premiers niveaux de maîtrise algorithmique correspondent aux points d'Xp de maîtrise algorithmique décrits dans le tableau 2.

Niveau de maîtrise algorithmique	**Points Xp de maîtrise algorithmique**	**Points Xp de maîtrise algorithmique cumulés**
Niveau 0	0 point	0 point
Niveau 1	4 points	4 points
Niveau 2	16 points	20 points
Niveau 3	36 points	56 points
Niveau 4	64 points	120 points
Niveau 5	100 points	220 points
Niveau 6	144 points	344 points
Niveau 7	196 points	540 points
Niveau 8	256 points	796 points
Niveau 9	324 points	1120 points
Niveau 10	400 points	1520 points
Niveau 11	484 points	2004 points
…	…	…

Tableau 2 : Relation entre les niveaux de maîtrise algorithmique
et les points d'Xp de maîtrise algorithmique.

Globalement, ainsi, les étudiants possèdent un niveau de maîtrise algorithmique qui correspond au niveau de difficulté algorithmique des exercices qu'ils savent résoudrent en général. Les exercices de niveau de difficulté algorithmique directement supérieur sont accessibles à l'étudiant et constituent des défis pour lui (voir la définition précise du niveau d'exercice accessible par un langage informatique spécifique), et les exercices de niveau de

difficulté algorithmique largement inférieur sont réputés déjà faits ou facilement faisables (la solution, lorsqu'elle existe, peut donc être décryptée), leur résolution n'apporte que peu de points.

Les règles d'accès aux exercices introduisent une progression de l'étudiant qui est organisée de la manière suivante :

- l'étudiant commence avec 0 point d'Xp de maîtrise algorithmique et peut donc accéder seulement aux exercices de niveau 0-10,
- à chaque résolution d'exercice (correcte, partiellement correcte, ou même très imparfaite), l'étudiant gagne des points d'Xp de maîtrise algorithmique,
- après avoir résolu quelques exercices correctement, l'étudiant a acquis des points d'Xp de maîtrise algorithmique et doit pouvoir aborder des exercices de niveau supérieur (en cas de présence de nombreuses erreurs dans les solutions, cela peut/doit demander quelques exercices de plus),
- dans tous les cas, l'étudiant doit pouvoir observer à chaque instant sa progression et la prochaine limite à atteindre.

Exemple de progression :

- état initial : 0 point d'Xp de maîtrise algorithmique, maîtrise niveau 0, niveau d'exercice accessible 10 ;
- résolution correcte d'un exercice de niveau 1 : gain de 1 point d'Xp de maîtrise algorithmique, 1 point d'Xp de maîtrise algorithmique en tout, maîtrise niveau 0, niveau maximal d'exercice accessible 10 ;
- résolution partielle d'un exercice de niveau 2 : gain de 1 point d'Xp de maîtrise algorithmique, 2 points d'Xp de maîtrise algorithmique en tout, maîtrise niveau 0, niveau maximal d'exercice accessible 10 ;
- résolution correcte d'un exercice de niveau 3 : gain de 9 points d'Xp de maîtrise algorithmique, 11 points d'Xp de maîtrise algorithmique en tout, maîtrise niveau 1, niveau maximal d'exercice accessible 11 ;
- résolution partielle d'un exercice de niveau 3 : gain de 4 points d'Xp de maîtrise algorithmique, 15 points d'Xp de maîtrise algorithmique en tout, maîtrise niveau 1, niveau maximal d'exercice accessible 11 ;
- résolution correcte d'un exercice de niveau 3 : gain de 9 points d'Xp de maîtrise algorithmique, 24 points d'Xp de maîtrise algorithmique en tout, maîtrise niveau 2, niveau maximal d'exercice accessible 12 ;
- ...

La méthode proposée pour l'évaluation des niveaux de difficulté algorithmique et de la progression des étudiants dans leur niveau de maîtrise algorithmique peut être modifiée ou simplifiée.

Pour motiver les utilisateurs et stimuler leur utilisation d'EDBA, EDBA 1.0 doit aussi pouvoir :

- donner un hit-parade des maîtrises algorithmiques, des gains de points d'Xp de maitrise algorithmique et des gains relatifs selon les langues et les langages, pour ceux que l'esprit de compétition attire ;
- donner les temps de résolution, les nombres d'exercices abordés, de tests effectués globalement par tous les utilisateurs d'EDBA pour le dernier jour, la dernière semaine, pour ceux que l'esprit de communauté motive ;

- donner un problème de la semaine (selon le niveau), un concours du mois, la liste des derniers problèmes, ou des problèmes sans solution, pour ceux qui aiment les défis, etc.
- posséder des actualités, des forums de discussions, etc.

EDBA 2.0 – Enrichissement participatif de la base d'exercices, portefeuille de droits&devoirs.

L'objectif visé par cette version est d'introduire des aspects « participatifs », « collaboratifs » dans la gestion et l'enrichissement de la base d'exercices. Le choix du numéro de version 2.0 n'est donc pas seulement lié au processus de développement, mais fait aussi référence aux notions habituellement associées au Web 2.0.

La mise en place de cette phase du développement d'EDBA a deux pré-requis. Le premier -d'ordre technique- est simplement lié au développement « normal » de l'application, i.e. l'introduction de nouvelles fonctionnalités, permettant la gestion des collaborations. S'il représente un travail significatif, ce pré-requis technique ne constitue pas un défi au sens des points techniques précédents. Le second pré-requis dépend des usages d'EDBA ; pour mettre en place EDBA 2.0, il faut que suffisamment d'exercices et/ou d'étudiants peuplent EDBA, afin qu'un travail de gestion participatif et collaboratif autour d'EDBA puisse avoir lieu et soit nécessaire.

La rupture avec EDBA 1.0, est produite par la disparition de l'organisation en deux classes étudiant-enseignant pour laisser la place à une seule catégorie d'utilisateurs associée à la base d'exercices d'algorithmique obtenue avec EDBA 1.0. Les membres de cette classe unique d'utilisateurs gagnent la possibilité, selon les compétences de chacun, de gérer cette base pour qu'elle puisse se développer. Ce travail participatif est organisé de manière collaborative.

Plus précisément, dans EDBA 2.0 il est prévu que les utilisateurs gardent un droit d'accès et de lecture de la base d'exercices limité par leur maîtrise algorithmique, comme dans EDBA 1.0, mais, ils peuvent également enrichir la base, participer à son organisation, l'améliorer, la critiquer, lever et gérer les conflits qui peuvent apparaître. Ces actions sont limitées par leur maîtrise de l'algorithmique et la connaissance d'EDBA acquise au cours de l'utilisation qu'ils en font. Ainsi, la pratique d'EDBA n'est pas seulement utile pour apprendre l'algorithmique, mais aussi pour améliorer et soutenir EDBA.

Ces démarches prennent la forme de droit et devoirs vis-à-vis de la gestion d'EDBA que nous notons Droits&Devoirs. L'acquisition de ces Droits&Devoirs est similaire à l'acquisition des points Xp de maîtrise algorithmique. Ces Droits&Devoirs concernent autant EDBA que l'algorithmique. Ils sont associés à des droits sur certaines actions de gestion d'EDBA, et devoirs en regard de ces droits. Il s'agit d'un accès orthogonal à EDBA.

La définition et l'acquisition de ces Droits&Devoirs est l'un des éléments caractérisant EDBA 2.0 ; la gestion des collaborations et conflits sera l'autre point important.

En regroupant les caractéristiques des utilisateurs d'EDBA 1.0, étudiants et enseignants, et en ajoutant les éléments nécessaires au travail participatif et au travail collaboratif, l'utilisateur d'EDBA 2.0 est identifié par son login et son mot de passe. Un nom réel, une adresse de courriel vérifiée sont associés ainsi qu'un profil public. Ce profil comporte (en gras les éléments nouveaux) :

- un texte libre (descriptif) (éventuellement en HTML) [optionnel]
- **l'appartenance de l'utilisateur à une liste de groupes d'utilisateurs [optionnel]**
- le lien vers une page web (extérieure au site EDBA) [optionnel]
- la liste des contributions de l'utilisateur
- le niveau de maîtrise algorithmique de l'utilisateur pour chaque langage informatique

- le niveau de maîtrise algorithmique absolue de l'utilisateur
- le niveau de difficulté algorithmique maximum d'exercice accessible de l'utilisateur
- le nombre de points Xp de maîtrise algorithmique absolue nécessaire pour accéder à un niveau de maîtrise algorithmique supérieur
- **un portefeuille de Droits&Devoirs pour le travail de gestion participative et collaborative de la base d'exercices, comptés chacun en points de Droits&Devoirs, parmi les Droits&Devoirs suivants :**
  - **droit&devoir dans le choix d'un nom d'exercice, et de l'arité du problème**
  - **droit&devoir pour rédaction de sujets informels d'exercices**
  - **droit&devoir pour rédaction de spécifications formelles d'exercices**
  - **droit&devoir pour rédaction de jeux d'essais**
  - **droit&devoir pour rédaction de solution dans un langage spécifique**
  - **droit&devoir pour fixer la difficulté algorithmique d'un exercice**
- **une option (opt-out) pour accepter, limiter, refuser les appels par courriel d'EDBA à ces Droits&Devoirs, pour participer à la gestion d'EDBA (gestion des conflits, appel à compléter, appels à valider un exercice, …)**

et plusieurs fonctionnalités (accessibles à l'utilisateur seulement) :

- l'accès aux travaux anonymés des autres utilisateurs sur les exercices introduits par l'utilisateur lui-même ;
- la possibilité d'envoyer un message aux autres utilisateurs de ces exercices (tout en conservant l'anonymat de ces utilisateurs) ;
- la sauvegarde des exercices travaillés (état courant, états significatifs -corrects syntaxiquement) et du temps de travail effectif passé sur chaque exercice ;
- l'utilisateur peut lever son anonymat via la garantie d'EDBA (c'est EDBA qui lève l'anonymat) et donner son nom réel et son adresse de courriel vérifiée à un autre utilisateur d'EDBA ;
- l'utilisateur peut voir, décrypter et utiliser les solutions des exercices ayant un niveau inférieur à son niveau de maîtrise algorithmique accessible - 8.

Les Droits&Devoirs et le portefeuille de Droits&Devoirs sont assimilables, respectivement, à la reconnaissance de compétences et à un portefeuille de responsabilités.

Le choix du terme « Droit&Devoir » plutôt que compétence tend à répondre à deux objectifs : d'une part éviter le terme « compétence » dont le champ sémantique est trop large et sujet à controverse (c'est pour une raison similaire que le terme point Xp de maîtrise algorithmique a été inventé à mi-chemin entre point d'expérience algorithmique ou point d'expertise algorithmique), d'autre part, si l'aspect « droit » (droit d'accès, par exemple) semble clair, il nous a semblé intéressant de rappeler qu'un droit trouve souvent son équilibre avec un devoir associé.

Cela demande peut-être un point d'éclaircissement. Prenant un peu de recul sur les organisations humaines, on constate qu'à partir du moment où l'on a la compétence pour réaliser une action, que l'action est en puissance entre nos mains, alors il est parfois nécessaire d'avoir également le droit de la réaliser pour que l'acte puisse se faire. Réciproquement, parfois, lorsque l'on a la compétence pour réaliser une action il est nécessaire d'utiliser cette compétence, à la demande d'un tiers, ou d'une situation. Dans ces conditions, la possession de la compétence nous impose le devoir de l'utiliser. C'est le cas, souvent, des experts, médecins par exemple.

Un terme plus complet « Droit&Compétence&Devoir » ou « Compétence&Droit&Devoir » aurait été possible s'il n'avait été un peu plus lourd, un peu trop lourd. Droits&Devoirs est déjà assez lourd. « D&D » serait peut-être à considérer s'il n'était trop vide de sens à sa naissance.

EDBA 2.0 – gestion participative et collaborative de la base d'exercices

L'évolution du portefeuille de Droits&Devoirs d'un utilisateur suit son niveau de maîtrise algorithmique et son utilisation d'EDBA. À chaque exercice résolu, il reçoit pour chacune des compétences distinguées dans EDBA autant de points de Droits&Devoirs que de points d'Xp de maîtrise algorithmique.

Ces Droits&Devoirs ne sont pas figés, ni fixés uniquement par le système. Ils peuvent évoluer au cours du temps selon les choix de l'utilisateur.

Parmi ces points de Droits&Devoirs, 90% des points d'un droit&devoir donnés sont liés initialement à ce droit&devoir, 10% peuvent être transférés par l'utilisateur de ce droit&devoir vers un autre droit&devoir. En cas de transfert, une perte de 25% de ces points est appliquée ; un gel pour un mois de la transférabilité de ces points est –par ailleurs- appliqué. Des mécanismes similaires sont en œuvre dans certains jeux sur internet.

Un mécanisme d'oubli des Droits&Devoirs rarement utilisés est aussi à prévoir. Un modèle utilisant les travaux de psychologie cognitive sur la mémoire peut être utilisé, par exemple, en suivant les systèmes de Flashcards [Leitner & Totter – 72].

Comme on l'a vu au début de cette section, ces Droits&Devoirs concernent la rédaction d'exercices, au sens large, découpés en éléments distinguant chacune des parties d'un exercice (l'énoncé, la spécification, les jeux d'essais, etc.).

Ce découpage a pour objectif de permettre des travaux collaboratifs entre individus ayant des compétences complémentaires dans différents domaines.

Ainsi, quelqu'un de littéraire pourra-t-il se charger de la rédaction de l'énoncé de l'exercice, quelqu'un de plus formel pourra rédiger les spécifications et quelqu'un de pratique pourra rédiger les jeux d'essai.

Des Droits&Devoirs concernent aussi la gestion des conflits qui risquent de survenir lors des travaux concurrents (la concurrence étant entendue au sens informatique, comme par exemple dans les systèmes de gestion de versions concurrentes [CVS], c'est-à-dire le travail en parallèle, qui est nécessaire à un travail collaboratif et qui mène à des modifications incompatibles –conflits, au sens informatique– et qu'il faut reconstruire ou fusionner « à la main » ; la concurrence au sens commun qui en découle et qui mène aux conflits –au sens commun– quand deux personnes ne sont pas d'accord et s'opposent sur la manière de rendre compatibles les différents points de vue et où alors une négociation est nécessaire, éventuellement sous le patronage d'un tiers qui encadrera la concertation et amènera chacun à faire peut-être quelques concessions pour arriver à un consensus)

Plus précisément, l'utilisation des Droits&Devoirs dépend du nombre de points de Droits&Devoirs obtenus par l'utilisateur et de la complexité algorithmique de l'exercice considéré. L'utilisateur peut utiliser un droit&devoir (rédiger/valider/gérer un conflit vis-à-vis d'un exercice/test/solution/spécification) pour des exercices ou proposition d'exercices complets ou incomplets selon les règles d'utilisation des droits&devoirs suivantes :

- pour lever des conflits : utilisation possible sur des exercices de difficulté algorithmique inférieure au niveau de droit&devoir acquis -3
- pour la rédaction d'éléments et l'appel à compléter : utilisation possible sur des exercices de difficulté algorithmique inférieure au niveau de droit&devoir acquis – 6,
- pour la gestion de conflits et la validation d'exercices : utilisation possible sur des exercices de difficulté algorithmique inférieure au niveau de droit&devoir acquis -12.

Pour les derniers niveaux 80-100 d'exercice, et la gestion d'EDBA, des règles spéciales sont à définir.

Exemple d'utilisation

Un utilisateur ayant un niveau de maîtrise algorithmique de l'ordre de 14 aura :

- accès à des exercices de difficulté algorithmique jusqu'à 20,
- possibilité d'une observation libre des solutions d'exercice de niveau 6 ou moins,
- un droit&devoir pour lever des conflits sur des énoncés d'exercices de niveau 11 ou moins,
- des droits&devoirs en rédaction d'exercices/de spécifications/de tests/de solutions pour des exercices de niveau 8 ou moins (rappel : une fois rédigés, ces exercices devront être validés. Cela pourra se faire par un utilisateur ayant un niveau 20 ou plus),
- des droits&devoirs en validation d'exercice ou résolution de conflit pour des exercices de niveau 2 ou moins.

Remarque : la rédaction d'un exercice, et en particulier la rédaction des exemples, doit pouvoir se faire via un environnement spécifique facilitateur, utilisant en particulier la console pour la rédaction et la mise au point automatique des jeux d'essai ainsi que la vérification de la cohérence avec le programme de solution proposé.

Associé à ces règles d'utilisation des droits&devoirs, EDBA 2.0 possède un moteur interne déclenchant l'appel aux droits&devoirs nécessaires. Ainsi, la présence d'un exercice complet mis en attente de validation provoque l'envoi d'un message à un utilisateur compétent pour le valider. Ce message est renouvelé tous les jours d'attente (ou toutes les semaines, selon la variable de temps adoptée par le système), jusqu'à validation, en changeant éventuellement le destinataire du message.

Pour un exercice incomplet, un appel à compléter un exercice peut être soumis à EDBA, qui alors relai cet appel auprès de 2 personnes compétentes pour compléter et valider cet exercice. Ce message est renouvelé tous les jours d'attente (ou toutes les semaines, selon la variable de temps adoptée par le système), jusqu'à complétude et validation.

Associées à ces appels d'EDBA, des listes d'exercices incomplets ou à valider doivent être également visibles aux utilisateurs d'EDBA pour permettre d'autres démarches plus spontanées.

L'utilisation de ces échelles de Droits&Devoirs est à mi-chemin entre ce que l'on trouve dans certains CMS pour les processus de validation des informations avant publication, les droits Unix et les droits dans les premiers systèmes de discussion sur internet, les IRC : Internet Relay Chat [IRC user level].

Coté collaboration, la gestion des contributions engendrées par ces droits dans EDBA est donc plus encadrée que dans Wikipédia [Wikipédia]. S'il est toujours assez facile pour l'utilisateur de faire des propositions libres d'exercices ou seulement d'énoncés d'exercices (le droit&devoir de rédaction d'une proposition est assez facilement accessible dans la mesure où la progression dans EDBA est suffisante), ces propositions ne sont prises en compte qu'après validation par des utilisateurs experts d'EDBA. L'expertise, quant à elle, est obtenue d'après les compétences réelles et observées des utilisateurs, et non pas données a priori par un administrateur omnipotent comme cela peut se faire avec certains wikis.

La gestion des conflits diffère également de celle observée dans Wikipédia. L'introduction de contenus dans EDBA étant sujete à une validation plus forte, les conflits sont à prendre en compte autrement, et leur gestion ne peut être confiée à tout utilisateur. Les conflits peuvent porter sur le nom de l'exercice, la rédaction des différentes parties de l'exercice, le niveau de

l'exercice, etc. Le résultat d'un conflit peut-être le retour à une version antérieure (comme dans wikipédia) ou la suppression de l'élément incriminé.

Une première limitation à l'utilisation des conflits est liée au niveau des exercices ; une autre concerne la fréquence d'utilisation de ce droit&devoir. Un utilisateur, dans la mesure où il en a la capacité, ne peut pas lever plus de deux conflits pour un exercice donné, sur une période d'un mois. La levée d'un conflit envoie un message à l'auteur de la partie incriminée et à un utilisateur compétent pour résoudre le conflit. Le message peut être renouvelé tant que dure le conflit.

L'exposé des mécanismes proposés pour le travail participatif, les collaborations et la gestion des conflits met en avant la nécessité pour l'utilisateur d'avoir les compétences nécessaires à ces activités pour pouvoir avoir le droit de les exercer. Il ne faudrait pas réduire ces compétences à des droits seulement. Cela n'apparait peut-être pas suffisamment dans l'exposé, mais ces compétences sont autant des droits que des devoirs. À un certain niveau de compétence, lorsque le droit&devoir de rédaction a été obtenu par un utilisateur *A*, mais que le droit&devoir de publier n'est pas obtenu par cet utilisateur *A*, le processus de publication du contenu produit par *A* passe par un appel à un utilisateur *B* ayant les compétences supplémentaires nécessaires à cette publication. Pour ce second utilisateur *B*, l'appel à validation venant de A via EDBA correspond à un usage de la compétence dû au système. Parce que *B* a le pouvoir d'utiliser sa compétence, le système se repose sur lui pour qu'il le fasse, et ce pouvoir devient un devoir. Ainsi, *A* peut voir son contenu publié.

Collaborations et conflits ne gèrent pas nécessairement tout ; par exemple, le niveau de difficulté d'un exercice est fixé initialement par l'utilisateur définissant son noyau (nom et énoncé informel) et éventuellement à travers un travail collaboratif.

Il peut évoluer par la suite après des phases de conflits/gestion de conflits.

Cette évolution peut aussi être le résultat d'analyses (automatiques) des résolutions de cet exercice par les utilisateurs de la base. Par exemple, la difficulté d'un exercice, son niveau, peut être défini à partir d'algorithmes de classification automatique, en prenant en compte son niveau initial (défini a-priori), le temps passé moyen pour la résolution de l'exercice, le nombre d'échec avant la résolution de l'exercice, le nombre d'échecs face à l'exercice (nombre de tests incorrects, avec leur degré de fiabilité), et tous autres éléments aussi objectifs que possible ressortant de la pratique d'EDBA. Un ordre non complet, sur la difficulté des exercices, peut également être considéré. (De même, le niveau des utilisateurs peut être calculé à partir d'algorithmes de classification automatique sur la pratique d'EDBA prenant en compte les exercices abordés, ceux réussis, les tests incorrects, ...)

En cas de changement du niveau d'un exercice, un message est envoyé aux auteurs des différentes parties de l'exercice (les points attribués pour la résolution de cet exercice seront ou ne seront pas rétroactivement modifiés).

D'autres possibilités de travail collaboratif plus classiques sont également introduites avec, par exemple, la notion de groupes de travail. Ainsi, l'espace de noms des fonctions doit pouvoir être organisé hiérarchiquement, avec en 1er un espace de noms spécifique pour l'utilisateur, venant ensuite des espaces de groupe et un espace public. L'objectif visé par cette hiérarchie d'espaces de noms est de permettre le partage de code et de solutions entre utilisateurs. Si une fonction ne se trouve pas dans l'espace de nom courant, elle peut être recherchée par élargissement successif, en respectant les niveaux de difficulté algorithmique, parmi les codes du groupes, puis les codes publics. Toutefois, l'accès à ces espaces est limité : une fonction de niveau N ne peut être accédée que par un utilisateur maîtrisant le niveau N+12 (par exemple, d'après ce qui précède).

À considérer également des écrans d'éditeur de code ou de console partagés pour le travail à plusieurs en même temps, et d'autres outils associés au travail collaboratif (forum, chat, …)

Au-delà, choix technologiques, conclusion et perspectives.

EDBA 3.0 Pour la recherche en didactique de l'algorithmique.
Au-delà d'EDBA 2.0, on peut imaginer un EDBA 3.0 centré sur des problématiques de recherche en didactique de l'algorithmique, orienté vers les chercheurs de cette discipline et les enseignants en informatique. Ainsi, divers outils pour l'enseignant et le chercheur ou didacticien en algorithmique peuvent être imaginés :

- diverses études des enregistrements anonymés
- outils de scénarisation et de visualisation des traces et contenus obtenus
- définition d'une plateforme de tests et d'expérimentation
- console de suivi des activités en direct ou en différé (magnétoscope)
- algorithmes de diagnostic des apprentissages observés
- modélisation sémantique des contenus
- …

Choix et défis technologiques
Les choix technologiques font rarement l'objet d'exposés approfondis, c'est bien dommage. Choisir une technologie n'est pas une chose anodine et est porteur de bien autant d'informations que certaines études statistiques trop locales [Trgalova&Al. - 09]. Ainsi, mes premiers projets [Bouhineau - 97], sous mac-OS, nécessitant un interpréteur spécifique (PrologIII, payant de surcroît) peuvent difficilement être comparés aux travaux entrepris depuis [Bouhineau - 03] pour Windows, sous forme d'exécutable distribué librement (un temps), sur le web, avec installateur et documentation.

La technologie envisagée pour EDBA :

- pour la couche métier, le client EDBA 0.0 (client lourd/riche) : javascript (ajax)
- pour les aspects BD : mysql
- avec un minimum de php pour faire la liaison entre javascript et mysql, et d'éventuels utilitaires (par exemple pour produire des graphiques, ou des pdf).

Divers formats d'exercices doivent être disponibles en sorties :

- format lisible humainement (pdf, html)
- XML, JSON

Ces choix technologiques sont portés par diverses raisons liées à la recherche et l'enseignement.

L'existence des plateformes d'enseignement, type Moodle, encourage l'écriture d'applications web reposant sur des technologies Ajax.

Les grandes possibilités d'échanges liée à la publication aisée des codes produits, les modèles ouverts (au sens open-source) pour les données favorisent la communication et le travail de recherche.

La légèreté des solutions logicielles mises en œuvre, les facilités d'installation (un navigateur et internet suffisent), les possibilités de paramétrisation, d'adaptation, de réutilisation, de localisation ou d'internationalisation permettent l'élaboration rapide de prototypes pour expérimenter ou essayer dans sa classe une version adaptée à son enseignement.

En particulier, pour les problèmes techniques, et les assemblages technologiques prévus, de gros espoirs sont fondés sur l'utilisation de javascript, le nouvel assembleur des applications web. Des interpréteurs d'une dizaine de langages informatiques existent déjà en javascript. Des éditeurs de code et des terminaux existent également. Javascript, à ne pas confondre avec java, est un langage plus puissant que sa réputation ne le laisse penser, pour le web et aussi pour un usage plus général. Il permet de faire des prototypes/expériences facilement. Une communauté importante utilise ce langage pour produire des solutions à de très nombreux problèmes, dont des problèmes non triviaux. Enfin, c'est un langage à la mode. Google, en particulier, l'utilise et cherche à le promouvoir activement.

Résumé des défis techniques prévus pour EDBA :

- application Ajax intelligente, c'est-à-dire obtenir une application comportant une intelligence mise en œuvre sur le poste client ;
- application Javascript + SQL, c'est-à-dire, limiter au strict minimum le nombre de langages nécessairse pour la réalisation de cette application web ;
- application Monopage [SPA], c'est-à-dire arriver à fusionner des codes ayant des origines et des méthodes de travail diverses ;
- application réseau et hors-réseau, c'est-à-dire concevoir une application qui puisse aussi continuer à fonctionner sans réseau (avec perte de certaines fonctionnalités, certes, mais permettant toujours de travailler sur un exercice obtenu avant le passage hors-réseau) ;
- développement comme pour les applications habituelles, c'est-à-dire utilisant les outils de génie logiciel usuels (« compilation » séparée, tests automatiques, séparation code-contenu-présentation, internationalisation, paramétrisation, gestion de la qualité du code, documentation du code, …) ;
- pouvoir ajouter un exercice en moins de 10 minutes.

Conclusions et perspectives

Le projet EDBA commencé en 2008 avec la rédaction de cet article (dans une forme plus réduite, certes, mais avant le début de la conception proprement dite et surtout, avant le début du codage effectif). Il a vu son développement et ses premières expérimentations en 2009 sur la base d'une application à mi-chemin entre EDBA 1.0 et EDBA 2.0.

Quelques résultats peuvent déjà être exposés informellement. La place prise par les jeux d'essai dans l'application semble intéressante, à la fois élément de réflexion et de compréhension de l'algorithme recherché et élément de validation des solutions proposées. Le mécanisme général d'attribution des points d'Xp de maîtrise algorithmique et le lien avec la complexité algorithmique des exercices a été bien perçu et est source effective de motivation chez les étudiants volontaires pour expérimenter de manière informelle l'application.
Voir http://www.noe-kaleidoscope.org/public/people/DenisB/EDBA/index_EDBA_Full.html pour la version courante.

À ce jour, pour l'essentiel, la plupart des défis techniques ont été relevés. Il reste cependant à affiner l'attribution des points d'XP de maîtrise algorithmique et l'organisation des droits&devoirs. Coté peuplement de la base d'exercices, 200 exercices de complexité algorithmique allant de 1 à 20, associés à plus de 1000 jeux d'essai, sont déjà disponibles. Comparé à ce que l'on trouve dans les livres d'enseignement de l'algorithmique, c'est un bon début, cf. Table 3. Une course au nombre est pour autant à éviter, d'autant plus qu'avec la multiplication des exercices, la nécessité d'une organisation plus fine de ces exercices commence à se faire sentir.

Les développements logiciels effectués ont provoqué un début de travail de réflexion vis-à-vis des tests et de la sécurité.

Sur les tests et jeux d'essai, la piste de tests aléatoires automatiques envisagée un temps a été mise en sommeil devant l'efficacité observée des tests choisis *à la main* comme ils ont été présentés dans cet article. La question de la validation des algorithmes n'est pourtant pas close. Une réflexion sur les outils à mettre à disposition des rédacteurs d'exercices pour choisir *à la main* ces tests est en cours. Une comparaison entre ces tests, choisis *à la main*, et les méthodes de correction manuelle (plus sémantique) est également en cours.

Vis-à-vis de la sécurité, deux sujets de préoccupation sont apparus : comment assurer la sécurité d'une base de données dans une application web accédée quasi directement via le client à travers javascript ? Comment limiter les tentatives de fraude d'étudiants malhonnêtes cherchant à piéger le logiciel en utilisant les possibilités de suivi d'exécution et de débuggage des codes javascript ? Des solutions partielles ont été trouvées. D'autres restent encore à trouver.

D'autres voies et perspectives de travail pour EDBA ont été imaginées, mais tout cela, ainsi qu'une analyse plus approfondie des premiers résultats obtenus avec EDBA et des enseignements tirés de la mise en œuvre effective de la version courante d'EDBA fera l'objet d'un autre article.

Manuel	Exercices/Problèmes	Algorithmes cherchés	Pages
Annales ENS Lyon/Ulm [Bougé & Al. 93]			143
Lyon	47	134	
Ulm	24	148	
Baynat [Baynat & Al. – 03]	144 exos, 722 Q	120	446
Flasque [Flasque & Al. – 10]	44	50	218
Knuth [Knuth – 98]			
T I	850	175	650
T II	900	123	762
T III	900	98	722
Cormen [Cormen & Al. - 10]			1146
parties 1-3	550	135	313
parties 4-6	838	266	366
annexes	89	1	66

Tableau 3 : Nombres d'algorithmes dans les exercices pour quelques manuels d'informatiques riches en exercices

Travail à faire / Sujets de stage possible (à part EDBA dans son ensemble) :

- **Interpréteurs & Co** : Rendre EDBA indépendant de la console, de l'éditeur et des interpréteur ou pouvoir être capable de proposer d'autres éditeurs, consoles et interpréteurs pour EDBA (en particulier des interpréteurs Ada, C, Java, ARM, CAML) ; et par suite répondre à la question suivante peut-on s'abstraire des langages informatiques ? Existe-t-il des adaptateurs entre langages (et structures de données) si l'on se restreint aux entrées/sorties ?

- **Test** : Validation d'algorithme par jeu de tests, génération de tests automatiques aléatoire ou selon des patrons symboliques par conparaison avec l'exécution d'une solution de référence, (avec test d'implication possible) et amélioration de la console pour la saisie de ces tests standards et symboliques

- Editeur : Éditeur de texte collaboratif (pour les 2 étudiants d'un binôme), et proposition d'un nouveau mode d'évaluation de l'expérience, de la maîtrise et des compétences d'un utilisateur

- Aide à la production d'algorithme à partir de généralisation d'actions effectuées dans une console via un interpréteur (forme de manipulation directe de suite d'actions), définition d'un langage de programmation par l'exemple et l'aide à la généralisation.

Contact : Denis.Bouhineau@imag.fr
Version : 1.1.8
Date : Octobre 2010

Références

[Baron et al. – 93] Baron, G.-L., Baudé, J., Bron, A., Cornu, P., Duchâteau, C. *Actes de la troisième rencontre francophone de didactique de l'informatique, SION Juillet 1992.* EPI (Association Enseignement Public et Informatique) (Ed.) 1993. http://edutice.archives-ouvertes.fr/edutice-00359968/en/

[Baron & Bruillard – 01] Baron G.-L., Bruillard E. *Une didactique de l'informatique ?* Revue française de pédagogie n° 135, avril-mai-juin 2001. http://edutice.archives-ouvertes.fr/edutice-00286326/fr/

[Baynat & Al. – 03] Bruno Baynat, Philippe Chrétienne, Claire Hanen, [et al.]. *Exercices et problèmes d'algorithmique : 144 énoncés avec solutions détaillées*. Dunod. 2003.

[Bougé & Al. – 93] Luc Bougé et Al. (ouvrage collectif) *Algorithmique : exercices corrigés posés à l'oral du concours d'entrée à l'Ecole Normale Supérieure de Lyon*. Ellipses-Marketing. 1993.

[Bouhineau – 97] Bouhineau D. *Construction automatique de figures géometriques et programmation logique avec contraintes*, Thèses. Université Joseph-Fourier - Grenoble I. 1997. http://tel.archives-ouvertes.fr/tel-00004922/fr/

[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, Environnements Informatiques pour l'Apprentissage Humain, Strasbourg, France, 15, 16 et 17 avril 2003. http://hal.archives-ouvertes.fr/hal-00190083/fr/

[Cormen & Al. 10] Thomas Cormen, Charles Leiserson, Ronald Rivest, Clifford Stein. *Algorithmique* Dunod. 2010.

[Dijkstra – 69] Dijkstra, E.W. *Notes on structured programming*. Technological University Eindhoven, Netherlands Department of Mathematics. 1969.

[Flasque & Al. – 10] Nicolas Flasque, Helen Kassel, Boris Velikson,... [et al.] *Exercices et problèmes d'algorithmique [Texte imprimé] : rappels de cours, exercices et problèmes avec corrigés détaillés, solutions en pseudo code et en langage C*. Dunod. 2010.

[Knuth – 98] D. Knuth. *The art of computer programming*. Addison-Wesley. 1998.

[Laborde - 88] *Actes du 1er colloque franco-allemand de didactique des mathématiques et de l'informatique,* Textes réunis et présentés par Colette Laborde, La pensée sauvage édition. 1988. ISBN : 2 85919 067 9.

[Libbrecht - 08] Libbrecht, P. *Cross curriculum search through the GeoSkills' Ontology*. Proceedings of the Second International Workshop on Search and Exchange of e-le@rning Materials (NL –

Maastricht), 2008. http://svn.activemath.org/intergeo/Papers/2008-SEAM-ontology-for-cross-curriculum/SEAM-ontology-for-cross-curriculum.pdf

[Leitner & Totter – 72] Leitner, S. and Totter, R. *So lernt man lernen*, Herder publisher, 1972.

[Lobato & Hache – 07] Lobato, Rafael et Hache, Sébastien. *Mathenpoche au fil des TICE*. MathémaTice N°6, Sept 2007. http://revue.sesamath.net/spip.php?article94

[Melis et al. – 01] Melis, E., Andres, E., Büdenbender, J., Frischauf, A., Goguadze, G., Libbrecht, P., Pollet, M. and Ullrich, C. *ActiveMath: A generic and adaptive web-based learning environment*. International Journal of Artificial Intelligence in Education. 12(4), 2001.

[Moreno-Navorro & Rodriguez-Artalejo – 92] Moreno-Navarro, Juan Jose, and Rodriguez-Artalejo, Mario. *Logic programming with functions and predicates: The language Babel.* The Journal of Logic Programming, 12(3), February 1992.

[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.

[Pears & Al. – 07] Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., and Paterson, J. *A survey of literature on the teaching of introductory programming*. SIGCSE Bull. 39, 4 (Dec. 2007), 204-223. DOI= http://doi.acm.org/10.1145/1345375.1345441

[Sanwin – 04] Sangwin, C. J.. *Assessing mathematics automatically using computer algebra and the internet*. Teaching Mathematics and its Applications, 23(1), 2004. http://web.mat.bham.ac.uk/C.J.Sangwin/Publications/tma03.pdf

[Trgalova&Al. - 09] Trgalova, Jana, Bouhineau, Denis and Nicaud, Jean-François, *An Analysis of Interactive Learning Environments for Arithmetic and Algebra Through an Integrative Perspective,* International Journal of Computers for Mathematical Learning 14(3), Springer, pages 299-331, 2009.

[Xiao – 01] Xiao, Gang. *WIMS: An Interactive Mathematics Server*. Journal of Online Mathematics and its Applications. 1 (1). Jan 2001. http://mathdl.maa.org/mathDL/4/?pa=content&sa=viewDocument&nodeId=354&pf=1

Webographie (par défaut, pages accédées en Juin 2010)

[ActiveMath] http://demo.activemath.org/ActiveMath2/main/menu.cmd

[CVS] http://www.nongnu.org/cvs/

[DADS] http://www.itl.nist.gov/div897/sqg/dads/termsType.html

[Elm-Art] http://art2.ph-freiburg.de/Lisp-Course

[IRC user level] http://www.ablenet.org/irc_user_levels

[MathEnPoche] http://mathenpoche.sesamath.net/

[Matexo] http://matexo.smai.emath.fr/

[Moodle] http://moodle.org/

[OpenId] http://openid.net/

[ProjectEuleur] http://projecteuler.net/

[P99] https://prof.ti.bfh.ch/hew1/informatik3/prolog/p-99/

[ShootOut] http://shootout.alioth.debian.org/

[Specif] http://www.specif.org/

[Spedago-lien inaccessible] http://spedago.unice.fr (lien inaccessible)

[SPOJ] http://www.spoj.pl/

[Wikipedia] http://fr.wikipedia.org/wiki/Aide:Sommaire
http://fr.wikipedia.org/wiki/Jeu_de_role_(jeu_video)
http://fr.wikipedia.org/wiki/Jeu_de_role_en_ligne_massivement_multijoueur
http://fr.wikipedia.org/wiki/Liste_des_algorithmes,
http://fr.wikipedia.org/wiki/Liste_des_langages_de_programmation
http://en.wikipedia.org/wiki/Spaced_repetition
http://fr.wikiversity.org/wiki/Faculte:Informatique

[WIMS] http://wims.unice.fr/wims/fr_home.html