FoKon un système d'éxamination automatique

Ivan Auge
and Oliver Pons

2008

Résumé

Cet article décrit Fokon, un logiciel qui automatise entièrement la correction, la notation et l'établissement de rapports d'examen. Ce logiciel est utilisé en production à l'ENSIIE depuis 2005. Le cahier des charges qui a conduit à sa conception ainsi que le fonctionnement global et la méthode de notation sont tour a tour présentés.

Le projet d'automatisation d'examens a commencé à l'ENSIIE en 2004 et répondait à un besoin de renfoncer le niveau de programmation moyen de nos élèves. Il a abouti à un logiciel appelé FoKon. Nous présentons d'abord le cahier des charges qui a conduit à sa conception puis nous détaillons le fonctionnement de l'outil.

1 Problématique générale

Lors de la réalisation de cet outil, il nous a fallu prendre en compte les problèmes principaux suivants.

Traces légales Un étudiant qui passe un examen doit pouvoir voir sa copie, et éventuellement contester sa note. Dans un établissement d'enseignement public, le sujet et les copies des examens doivent être archivés et conservés. Le support physique de ces copies peut être informatique.
Un système d'examens informatiques se doit de générer une copie avec le sujet pour chaque étudiant. L'archivage des copies peut être fait après l'examen manuellement et n'est donc pas du ressort du système d'examen.

La fraude Dans un examen classique sur table, il y a deux types principaux de fraude, la copie sur les voisins et, la copie sur des antisèches. Dans la version informatisée, ces possibilités de fraude sont encore plus grandes. En effet, un tel examen va se dérouler dans une salle classique de travaux pratiques où les postes de travail sont des machines en réseau. Ceci induit :

- Sur un écran, un utilisateur peut très facilement changer la taille de la police de caractères et changer légèrement l'orientation de son écran pour permettre à ses voisins proches de copier.
- Les machines étant en réseau, il peut transmettre ou donner accès à sa copie à d'autres étudiants. La notion de voisins n'existe en fait plus. Il peut éventuellement de la même manière transmettre le sujet à une tierce personne compétente et attendre que celle-ci lui retransmette la réponse.

— Les machines étant en réseau, il peut placer des antiséches quelque part sur le réseau et aller les consulter tranquillement.

Correction & domaine Dans ce projet, nous visions des examens avec correction automatique. Il est évident qu'une dissertation de philosophie est hors de ce cadre, et qu'un QCM (Questionnaire à Choix Multiples) y répond très bien puisque les réponses sont connues a priori. Entre ces deux extrêmes, nous avons choisi tout examen dont la réponse est l'écriture d'un ou plusieurs exécutables, ce qui couvre un bon nombre d'enseignements pratiques dans une formation en informatique.

2 Description

2.1 Vue globale

Le schéma général de l'application est donnée sur la figure 1. En a) Un enseignant décrit l'examen (voir la sous section suivante) et un compilateur génère les deux exécutables : *lanceur* et *joueur*. En b), un poste de travail est configuré en utilisant ces deux exécutables et un fichier de configuration (*scénario* sur la figure) qui donne les paramètres globaux de l'examen : nombre de questions, le barème, sa durée, où doivent-être envoyées les copies. Toutes les routes réseau peuvent être fermées sauf celles donnant l'accès aux deux serveurs. En c) le diagramme de phase indique comment se déroule un examen pour un étudiant arrivant sur son poste de travail.

repos Dans cet état l'exécutable *lanceur* propose une fenêtre de login classique et attend un couple (nom,mot-de-passe).

init Dans cet état après une vérification du couple (nom,mot-de-passe) de manière analogue au login standard, il enchaîne les actions suivantes :

1. Création du répertoire HOME.
2. Initialisation de l'examen pour l'étudiant. Ceci consiste à créer dans son HOME une arborescence appartenant à "root" décrivant l'examen en fonction du fichier scénario. Concrètement, il y a un fichier par question. Les questions peuvent être tirées au hasard parmi un groupe de questions ou être paramétrées. Dans ce dernier cas, les paramètres sont tirés au hazard également. De ce fait, la probabilité que deux étudiants voisins physiquement aient le même sujet est assez faible.
3. Lancement du programme *login* original avec le couple (nom,mot-de-passe), pour mettre l'étudiant dans un environnement Unix standard.
4. Programmation d'un timer sur la durée de l'examen et attente du *time-out*.

jeux Dans cet état, l'étudiant lance la commande *joueur* autant de fois qu'il le désire. A chaque fois, la commande *joueur* affiche l'énoncé de la question courante, elle corrige la question courante indiquant clairement ce que fait son programme et ce qui était attendu sur le premier jeu d'essai qui n'a pas fonctionné correctement et, elle affiche la note qu'il a actuellement à toutes les questions ainsi que le temps restant.

La commande *joueur* passe automatiquement à la question suivante dès qu'une note de 15/20 est atteinte, une option permet à l'étudiant de revenir à une

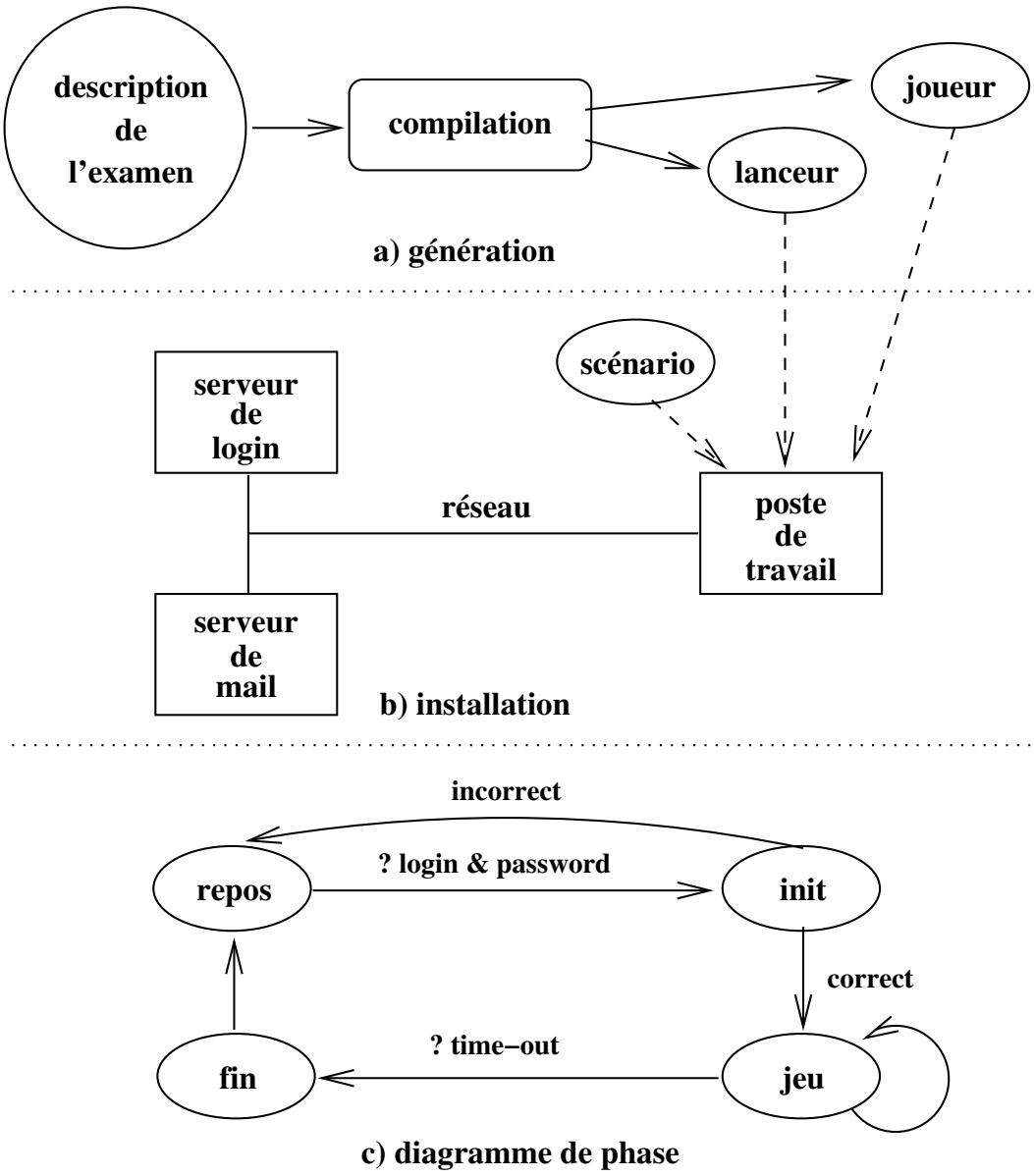

FIGURE 1 – Schéma général du système d'examen automatique.

question ou de traiter les questions dans l'ordre qu'il désire. Enfin, toute note obtenue à une question ne peut qu'être améliorée.

Au niveau interne, la commande *joueur* n'utilise que les fichiers créés par *lanceur* dans la phase **init**. De plus elle journalise chaque lancement que l'étudiant fait.

fin Dans cet état, le programme *lanceur* reprend la main. Il délogue l'étudiant de la station de travail, construit un e-mail. Le sujet de cet e-mail contient les informations générales (le nom de l'étudiant, le nom de la matière, la date, les heures de début et de fin et la note). Le corps de cet e-mail est une archive compressée du HOME de l'étudiant. Cette archive contient la production de l'étudiant, le sujet et le journal de son activité.
Cet e-mail est envoyé aux serveurs de mail mentionné dans le fichier *scénario*.

L'organisation logiciel assure un certain niveau de sécurité. D'abord, toute la phase d'examen (phase **jeu**) se déroule complètement localement sur le poste de travail. Une fois qu'un étudiant est logué, rien ne peut perturber l'examen si ce n'est une panne matérielle de son poste. Ensuite le poste de travail peut être quasiment coupé du reste du réseau, ce qui limite les possibilités de fraude. Enfin, la possibilité d'avoir des sujets différents par étudiant empêche la fraude sur les voisins proches.
Les obligations légales sont tenues par le e-mail, notons que si en fin d'examen cette procédure n'a pas marché, les données sont encore sur le disque de chaque station de travail et que l'on peut passer ave une clef USB pour les récupérer.

2.2 Correction automatique

Pour définir une question dans ce système d'examen, il faut fournir à la base deux fonctions. La première est une fonction de création, elle est appelée par la commande *lanceur* pour créer le fichier instanciant celle-ci. La seconde est la fonction de notation, elle est utilisée par la commande *joueur* pour noter la réponse à la question et générer le journal. L'API procure une bibliothèque pour implanter ces fonctions. Elle contient entre autres des routines pour lancer des exécutables ou des fonctions et récupérer leurs codes de retour ainsi que ce qu'ils écrivent sur les fichiers standard, des fonctions de comparaison implantant des filtres pour mesurer la divergence entre les résulats de 2 programmes.

Pour les questions ayant un seul exécutable comme réponse, la question peut être définie dans une structure XML et un analyseur XML génère les 2 fonctions. Les informations que l'utilisateur doit fournir dans cette structure sont :

l'énoncé C'est un texte définissant le programme à réaliser.

les arguments Ce sont le nombre d'arguments et leurs types, les types supportés étant : une énumération, une chaîne de caractère, un entier, un nom de fichier.

le programme de référence C'est un programme qui est la correction à la question. Il peut prendre la forme d'un source C, LEX, YACC ou LEX+YACC.

le nombre de jeux d'essais aléatoires Pour chaque paramètre du programme, un tirage aléatoire est effectué afin de générer un jeu d'essai. Ce tirage aléatoire peut être restreint de différentes manières. Par exemple, pour un argument qui est un fichier, il est possible de préciser son contenu en indiquant qu'il doit contenir 10% d'un certain mot et 90% d'un autre.

les jeux d'essais fixes Le jeux d'essais est complètement défini par l'enseignant. Ceci est utile, pour tester les cas extrêmes.

les filtres de comparaisons l'idée de base est de lancer pour chaque jeu d'essai le programme de l'étudiant et le programme de référence et de comparer les résultats. Les filtres possibles sont énumérés ci-dessous et ils peuvent être désactivés sauf les 2 premiers.

1. exactitude : les 2 résultats sont rigoureusement les mêmes.
2. exactitude lâche : les 2 résultats sont identiques si on ne tient pas compte des caractères de séparation (blanc, tabulation, ...).
3. inclusion : le rapport $\frac{A_N}{B_N}$, N étant un paramètre du filtre, A étant le nombre de chaînes de N caractères du résultat de référence qui sont présentes dans le résultat de l'étudiant, B étant le nombre de chaînes de N caractères du résultat de référence.
4. inclusion2 : le rapport $\frac{A}{B}$, N étant un paramètre du filtre, A étant le nombre de chaînes de N caractères du résultat de l'étudiant présentes dans le résultat de référence, B étant le nombre de chaînes de N caractères du résultat de l'étudiant.
5. code retour : comparaison des codes retours des deux programmes.

La note pour un jeu d'essai est produite en combinant les résultats de ces filtres, et la note à la question est obtenue en faisant la moyenne des notes de tous les jeux d'essai.

3 conclusion

Conçu début 2005 ce système, est à l'ENSIIE couplé avec YaKa [2] notre gestionnaire de parc pour installer rapidement un système à réseau réduit puis restaurer rapidement un système linux pleinement fonctionnel. Dans le courant de l'année 2005 il a été testé en grandeur nature (plus de 100 étudiants et 4 salles de TPs) pour des examen de programmation C en $1^{ère}$ année. L'expérience ayant été concluante en 2006, nous avons ajouté une épreuve de compilation (LEX/YACC) et une de middleware (RPC & CORBA) pour les étudiants de $2^{ième}$ année. Ces 3 épreuves ont lieu chaque année depuis plusieurs fois dans l'année. Les étudiants disposent également d'une version d'entraînement copie de la version d'examen mais avec des exercice devenus "un peu trop connu" et dont les traces ne sont pas envoyée.

L'utilisation de FoKon en production à l'ENSIIE depuis 2005 a démontré sa fiabilité et sa simplicité d'utilisation. Il a en outre permit d'améliorer sensiblement le niveau moyen de nos élèves en programmation.

Enfin FoKon est distribué sous license GNU sur le site [1] qu'il pourra ainsi être enrichi de nouveau exercice et de nouveau type d'épreuve.

Références

[1] The Fokon team. `http://yaka.ensiie.fr/fokon`.

[2] The Yaka Team. YaKa : An Environment For Describing And Installing Site Wide Systems. `http://yaka.ensiie.cnam.fr/`.