Le génie logiciel par la pratique : une expérience de projet en phases

Eric Cousin, Siegfried Rouvrais

Le génie logiciel par la pratique : une expérience de projet en phases

Éric COUSIN et Siegfried ROUVRAIS
École Nationale Supérieure des Télécommunications de Bretagne
Technopôle de Brest Iroise, CS 83818 - 29238 BREST CEDEX 3, France
e-mail : eric.cousin@enst-bretagne.fr, siegfried.rouvrais@enst-bretagne.fr

***Résumé*— Le génie logiciel constitue un enseignement difficile à faire passer auprès des étudiants sous forme magistrale. Les activités projet de programmation sont bien plus pertinentes pour son assimilation. Cet article décrit une articulation particulière de projets de développement logiciel, basée sur une succession de phases de modification de code. Outre les avantages habituels de la formation à l'ingénierie logicielle par projet, cette approche innovante permet d'une part de sérier les difficultés conceptuelles et techniques, et d'autre part d'aborder la partie aval du cycle de vie du logiciel - maintenance, évolutions - qui n'est traditionnellement pas expérimentée par les étudiants. En outre, cette mise en perspective renforce la sensibilisation aux impératifs de qualité logicielle des phases amont - modélisation, conception et codage. Différents aspects de cette forme originale de projet sont présentés, et un premier bilan d'expérience est tiré.**

***Mots clés*— Pédagogie par projet, conduite de projet de programmation, génie logiciel, cycle de vie aval du logiciel, travail collectif, autonomie, initiative, encadrement.**

I. INTRODUCTION

En informatique, le génie logiciel [5] consiste à guider les pratiques d'*ingénierie* afin de *produire et maintenir* des logiciels respectant un compromis de qualités, de délais et de coûts. Parce qu'elle traite plus des aspects organisationnels que purement techniques, cette discipline encore jeune constitue un enseignement *difficile à faire passer* auprès des étudiants en informatique. Perçues le plus souvent comme « allant de soi » lorsqu'elles sont présentées sous forme magistrale, les bonnes pratiques d'ingénierie logicielle sont plus favorablement abordées par une *mise en situation* à travers des projets de développement logiciel.

Le plus souvent, la réalisation d'un projet de programmation dans le cadre scolaire s'axe sur les activités « amont » du cycle de vie du logiciel (c.-à-d. la production) telles que la spécification, la modélisation, la conception et le codage. Les activités « aval » du génie logiciel (p.ex. *maintenance*, révision, *évolution* des logiciels) ne sont par contre que rarement abordées dans la pratique [6], alors qu'elles constituent une *part importante* de l'activité des métiers du logiciel. Ceci est d'autant plus dommageable que ce sont ces mêmes activités qui justifient le mieux les *impératifs de qualité* requis en amont (p.ex. structuration, modularité, documentation).

Cette année, dans le cadre de la formation de mastère « Ingénierie des Systèmes Informatiques Communicants » (ISIC) de l'ENST Bretagne, nous avons mis en place une *nouvelle activité collective* de développement logiciel, initiée en début d'année scolaire et menée à son terme fin mars 2003. Cette activité est une évolution d'une expérience précédente, l'atelier logiciel [2], menée depuis plusieurs années dans notre établissement. Tout en essayant de garder les atouts en terme de *pédagogie par projet* de cet exercice, notre nouvelle activité, baptisée « fil rouge », vise à aborder les *aspects avals* du cycle de vie logiciel, en confrontant des équipes d'étudiants à des scénarios d'évolution de leur logiciel. Nous nous appuyons en cela sur des travaux menés par ailleurs sur la formation à l'ingénierie logicielle collaborative et inspirés des pratiques de la communauté du logiciel libre [3], [4].

Dans un premier temps, cet article rappelle quelques acquis de l'activité atelier logiciel. Il décrit ensuite la nouvelle activité fil rouge, telle qu'elle était prévue en début d'année, et ses objectifs pédagogiques. Son déroulement effectif est ensuite présenté et les auteurs proposent un retour d'expérience sur l'activité, riche en terme de réflexions et pratiques pédagogiques. Finalement, avant de conclure, des perspectives d'amélioration de l'activité sont discutées à travers un premier bilan.

II. L'ATELIER LOGICIEL

L'atelier logiciel [2] était un exercice de travail en équipe dans un temps très fortement contraint (période bloquée de quelques jours) dont le principal objectif était de faire vivre aux étudiants les différentes composantes d'un projet de *développement logiciel*, et de les confronter à la prépondérance des *facteurs humains*. Pour les étudiants de profil informatique[1], il s'agissait d'élaborer une réponse à un appel d'offre, de développer un logiciel prototype et de présenter le tout en séance plénière. Parallèlement, une démarche réflexive sur le fonctionnement des équipes était demandée.

Les principales conclusions que nous avions tirées de cette activité pédagogique étaient les suivantes :
- l'intérêt principal de l'exercice réside dans son *intensité*. Malgré une faible intrusion dans l'emploi du temps, il permet d'illustrer de nombreuses facettes de la vie d'un projet ;
- le temps imparti ne permet pas d'atteindre des objectifs pédagogiques très ambitieux pour ce qui est de la

[1] L'atelier logiciel concernait également des étudiants d'options moins techniques.

gestion de projet; en revanche, l'atelier logiciel permet judicieusement de mettre en pratique diverses *connaissances techniques* enseignées dans le cursus, et peut de ce fait servir de gros travail pratique intégrateur ;
- l'atelier logiciel est en général bien *apprécié des étudiants*, comme en témoigne leur fort engagement. Ceci est lié à leur préférence pour les activités pratiques et à l'aspect « défi »,
- l'atelier logiciel participe à la création d'une certaine cohésion dans la promotion d'étudiants ;
- c'est sans doute la *capacité à travailler en équipe* qui est la plus développée à travers cet exercice ; la *démarche introspective* demandée à chaque équipe est à ce titre très formatrice ;
- les étudiants sont demandeurs d'une *appréciation qualitative* rapide sur leur travail ;
- vue la brièveté de l'exercice, une attention particulière doit être accordée à son *organisation logistique* ; plusieurs dispositions et astuces permettent d'assurer un bon déroulement.

Plusieurs points n'avaient pas été tranchés :
- est-il souhaitable de donner aux étudiants toutes les informations techniques nécessaires *avant* l'atelier ?
- est-il souhaitable de former les équipes plus tôt et de les former à des outils et méthodes de développement, leur donnant ainsi une sorte de *culture d'entreprise* ?

III. LE FIL ROUGE

Pour l'année scolaire 2002-2003, nous avons souhaité rénover fortement l'atelier logiciel, en tenant compte des bilans qui en avaient été tirés et des souhaits qui avaient été exprimés par les étudiants. Cette évolution, que nous envisagions depuis deux ans, a été rendue possible par une refonte de l'emploi du temps : il nous était donc possible de prévoir pour le fil rouge plus de temps que les quelques jours de l'atelier logiciel.

A. Descriptif

Le fil rouge est un travail d'équipe[2] qui s'*étale* tout au long du cursus ISIC (c.-à-d. un semestre). Il est scindé en plusieurs « phases ». Une application de base est d'abord réalisée (développement logiciel) par les équipes, puis *enrichie* plusieurs fois, lors des phases successives, pour répondre à de *nouveaux besoins*. Cette approche permet de prendre en compte les nouvelles compétences développées par ailleurs dans le cursus.

Chaque phase commence par la présentation, par les encadrants, du travail à effectuer et comprend un certain nombre de demi-journées de travail autonome. Pour chaque équipe, de trois à quatre étudiants, une phase se termine par la *remise d'un délivrable* (c.-à-d. documents, code source, carnet de bord) et une présentation en séance plénière du travail effectué. À titre d'exemple, le fil rouge 2002/2003 a consisté en l'élaboration d'un *agenda partagé* (p.ex. gestion de planning, enregistrement de tâches, rendez-vous) en langage de programmation Java. Pour économiser le temps passé dans les phases amont de la production, nous avions directement fourni aux équipes un cahier des charges contenant une pré-analyse du problème (c.-à-d. spécifications et modélisations sous la forme de diagrammes UML). Ce cahier des charges était non contractuel et pouvait supporter des mises à jour si elles étaient justifiées. Le scénario proposé était le suivant :
- phase 1 : élaboration du noyau de l'application,
- phase 2 : extension par connexion à une base de données,
- phase 3 : ré-ingénierie de la partie interface homme-machine (IHM ou interface graphique),
- phase 4 : découpage et distribution de l'application sur plusieurs machines (aspects multi-utilisateur).

Chaque équipe dispose d'un *interlocuteur privilégié* déterminé parmi les encadrants. Différents rôles « tournants »[3] sont définis librement dans chaque équipe, mais chaque membre est censé avoir connaissance de tout ce qui est fait par ses collègues. *À la fin du projet, chaque membre doit ainsi être capable de réaliser seul l'ensemble du travail.*

Pour conserver les avantages identifiés dans l'atelier logiciel, en particulier ceux liés à la *densité de l'effort*, une ou deux des quatre phases du fil rouge doivent être programmées sur une *courte période bloquée*. Un travail introspectif sur le fonctionnement de chaque équipe est également prévu. Nous pouvons donc considérer que le fil rouge englobe l'ancienne activité atelier logiciel.

B. Objectifs pédagogiques

Les objectifs pédagogiques du fil rouge sont multiples.

Tout d'abord, nous cherchons à conserver les principaux apports pédagogiques de l'atelier logiciel : appréhender les différentes facettes d'un projet et apprendre à *travailler en équipe*. Le déploiement de l'activité sur l'ensemble du semestre se prête par ailleurs bien au développement d'une mini *culture d'entreprise* au sein des équipes.

Comme c'est le cas pour de nombreuses activités de projet, le travail demandé permet de mettre en pratique différents savoirs et savoir-faire abordés préalablement dans le cursus (p.ex. la programmation en Java, l'utilisation de bases de données, le développement d'applications distribuées). L'avantage du découpage du projet en phases est de permettre une assimilation progressive de chacun des aspects techniques tout en gardant une vision systémique de l'ensemble et en aboutissant à un produit d'une taille et d'une envergure intéressante (et donc motivante).

La succession de phases ouvre par ailleurs la voie à un effet de *pédagogie active* [1] autour du génie logiciel, et c'est là une nouveauté, où les étudiants sont parfois *confrontés*

[2] Notons que dans le cadre du mastère ISIC, les étudiants réalisent également un projet de développement individuel.

[3] À titre d'exemple : chef de projet, testeur, intégrateur, responsable document, présentation.

à des situations ou problèmes *avant d'avoir les connaissances* ou compétences pour les résoudre correctement. Il en est ainsi, dans notre exemple, de la mise en œuvre de la partie IHM : ce qui a été réalisé lors de la phase 1 - avant les enseignements IHM - est critiqué en début de phase 3 - après l'enseignement IHM - et est remanié lors de la phase 3. Notre conviction est qu'il s'agit là d'une très bonne façon d'aborder l'enseignement du génie logiciel : le fait de reprendre, modifier/adapter à chaque phase ce qui a été fait dans les phases précédentes illustre particulièrement bien les bonnes pratiques d'ingénierie logicielle que nous voulons promouvoir (c.-à-d. documentation technique, modularité, prototypage, gestion de versions, etc.) et permet de préparer l'acquisition de nouvelles compétences dans ce domaine. Certains de nos enseignements - sous forme classique - de génie logiciel sont ainsi programmés *après* que les étudiants aient été confrontés aux problèmes correspondants.

Enfin, comme expliqué en introduction, les différentes évolutions du logiciel réalisé permettent d'illustrer la partie aval du cycle de vie du logiciel, et permettent d'apporter aux étudiants une vision plus large et réaliste des activités liées au développement logiciel. Ce n'était pas le cas avec l'atelier logiciel, ou avec un projet classique.

IV. DÉROULEMENT EFFECTIF

C'est la première année que cette activité était menée. Il s'agissait pour nous d'une *expérimentation pédagogique*, et les étudiants en étaient avertis. Malgré une préparation imparfaite, elle a dès le début reçu un excellent accueil des étudiants. Trop, peut-être : dès la première phase, nous avons constaté qu'ils investissaient *énormément de temps et d'énergie* sur cette activité, beaucoup plus que nous l'escomptions.

Objectivement, le volume horaire imparti pour la première phase avait été sous-dimensionné. En effet, la maîtrise des outils de modélisation et de programmation faisait défaut à la plupart des équipes. De plus, très peu des étudiants étaient sensibilisés au *travail en équipe*. La mise en jambe s'est donc avérée coûteuse en temps. Surtout, très motivés par l'activité et en situation d'*autonomie*, les étudiants se sont parfois engagés dans des *directions non indispensables* pour le produit (mais souvent formatrices). Sur une prévision initiale d'une quinzaine d'heures par étudiant, la phase 1 s'est largement étendue à une cinquantaine d'heure de travail effectif par étudiant, et ce au détriment des autres activités du cursus (p.ex. cours, travail personnel).

Nous avons donc pris la décision d'ajourner le fil rouge dès la fin de la première phase, pour permettre aux étudiants de suivre le cursus dans de bonnes conditions. Ce n'est que sur les trois dernières semaines du semestre qu'a été programmée la fin du fil rouge, à plein temps. Il n'y avait alors que le temps de faire deux phases, au lieu des trois prévues initialement ; après avoir traité les aspects base de données (phase 2), les étudiants avaient donc le choix entre traiter les aspects IHM (phase 3) ou les aspects distribution (phase 4).

Cette première occurrence du fil rouge ne respecte donc pas à la lettre le déroulement qui était prévu. Elle en a tout de même gardé l'esprit, comme nous allons le voir maintenant.

V. RETOUR D'EXPÉRIENCE

Globalement, l'activité a été très appréciée des étudiants et leur a permis - de par l'envergure technique du produit réalisé - de se sentir mieux « armés » pour leur très prochaine activité professionnelle.

Les étudiants étaient répartis en cinq équipes de trois à quatre membres. Ces étudiants proviennent d'horizons assez différents, et leur profil en terme de compétences informatiques (théoriques et techniques) est donc assez hétérogène en début de cursus, même après une période d'harmonisation intensive. En ce sens, le travail en équipe a largement favorisé le *partage de compétences* et a joué son rôle *intégrateur*.

À la fin de chaque phase, chaque équipe présentait et justifiait en session plénière ses propositions et productions et rendait un document décrivant la proposition. Ce *suivi* nous a permis de garder un contact avec les équipes tout en leur imposant et validant des échéances. Les présentations, de l'ordre d'une vingtaine de minutes par équipe et par phase, ont permis de *confronter les solutions* à celles des autres équipes et ont ouvert la voie à des questions/réponses critiques entre équipes. Cette approche a engendré une *émulation* entre les différentes équipes ; aux dires des étudiants, celle-ci s'est quelque peu atténué sur la fin.

Nous décrivons ci-après le déroulement des différentes phases et des approches et apports associés.

A. *Phase 1 : élaboration du noyau de l'application*

Cette phase s'est déroulée entre octobre et décembre. Les équipes disposaient d'une pré-analyse (éléments des premières phases amont de la production logicielle) fournie par les encadrants. Celle-ci était partielle et ouvrait ainsi la voie à la critique et l'extension afin de *favoriser l'esprit d'initiative* des différentes équipes. Après le temps requis pour s'imprégner des données du problème, cette phase a permis aux équipes de :

1. critiquer la pré-analyse fournie,
2. produire de nouvelles idées de conception,
3. confronter les points de vue des différents membres,
4. se concerter sur les alternatives de conception,
5. se confronter aux problèmes d'organisation inhérents au développement collaboratif, en particulier s'impliquer dans la planification des tâches (c.-à-d. division/répartition, estimation du temps de développement, métrique pour quantifier ce temps, respect des délais),
6. organiser les phases de test pour la validation de la production.

À l'issue de cette phase, les équipes disposaient d'une première version de leur produit.

Outre les habituels intérêts pédagogiques liés à l'activité de développement logiciel en équipe, notons que l'approche par phases a porté ses fruits dès cette première phase. Certaines équipes avaient ainsi déjà *anticipé* (ou plutôt « tenté d'anticiper ») les phases ultérieures dans leur conception initiale (sans toutefois pouvoir le faire complètement puisque n'ayant pas encore les compétences nécessaires).

De même, des *interrogations* sur les méthodes ou outils à adopter pour gérer le développement en équipe sont apparues *spontanément* et ont été soulignés lors des présentations de cette phase. L'approche a ainsi permis d'inciter les étudiants à l'utilisation d'outils d'aide au développement logiciels [4]. L'intérêt des enseignements correspondants, intervenus - à dessein - un peu plus tard, a été de ce fait *mieux perçu*.

B. *Phase 2 : connexion à une base de données*

Rappelons que la phase 2 avait été décalée vers le fin du semestre, début mars. Trois mois s'étaient donc écoulés depuis la fin de la phase 1, ce qui n'a pas manqué de poser quelques problèmes - *formateurs* - aux étudiants pour *reprendre* leur travail dans l'état où il l'avait laissé.

La phase 2 portait sur l'extension de l'application afin de rendre les données persistantes (p.ex. horaires de réunions, demande de rendez-vous). En fonction des divers enseignements dispensés auparavant dans la formation sur les bases de données, différentes alternatives s'offraient aux équipes qui n'ont pas toutes suivi les mêmes choix.

Sur cette phase, à *durée restreinte* par rapport à la première (une vingtaine d'heures en moyenne par étudiant), les rôles ont été beaucoup plus rapidement attribués et l'organisation plus efficace. Parfois, le rôle de chef de projet a été presque inexistant, se diluant sur l'ensemble de l'équipe. Certaines équipes ont axés leurs efforts sur les étapes de modélisation quant à leur ajouts/extensions plutôt que de s'engager directement dans la mise en œuvre. Au regard du temps imparti, leur produit final s'en trouvait ainsi plus limité mais leurs choix de conception plus clair et mieux justifiés. Les présentations finales ont ainsi mis en avant les *variétés* d'« attaques » du problème posé.

C. *Phases 3 et 4 : ré-ingénierie de la partie IHM ou distribution de l'application*

Suite aux problèmes de calendrier, les phases 3 et 4 avaient été mise au libre choix des équipes. Le travail présenté était axé sur les *justifications* plutôt que sur la réalisation informatique. En ce sens, quelques équipes se sont trouvées en quelque sorte frustrées de ne pas avoir un produit final opérationnel.

Sur cette phase, très dense et en toute fin de cursus, la plupart des équipes ne se sont pas vraiment attribué de rôles particuliers, d'autant plus que les membres n'étaient pas toujours pleinement disponibles aux horaires impartis (p.ex. entretien pour stages). La force de travail fut environ de 50h par équipe.

D. *Apports pédagogiques*

Étant donnés les bouleversements apportés à l'organisation de l'activité, nous n'avons pas de mesure objective de l'impact du fil rouge sur l'acquisition de compétences des étudiants. Notre jugement se base donc essentiellement sur les différents *debriefings* qui ont jalonné l'activité et sur les présentations de fin de phases.

Comme cela était le cas pour l'atelier logiciel, les étudiants conviennent que l'exercice leur apprend fortement à s'*organiser*, en particulier à prendre des *décisions collégiales* (ce qui explique la disparition du rôle de chef de projet pour certaines équipes). L'activité *illustre* bien les différentes facettes de la *vie d'un projet* et a donné aux étudiants une bonne occasion de *mettre en pratique* de nouvelles connaissances.

Le découpage en phases, même s'il n'a pas été mené comme prévu, a eu un apport indéniable. Les étudiants trouvent très formateur de *restructurer* plusieurs fois leur code, et ils apprécient d'avoir été en mesure de *réutiliser* des portions de code (classes) qu'ils avaient écrites dans les phases précédentes.

Nous notons également une nette amélioration de la *qualité des présentations* successives.

De façon générale, nous regrettons cependant que les phases aval du cycle de vie logiciel (p.ex. gestion des tests ou des différentes versions du logiciel) n'aient pas encore été assez abordées. Cette carence s'explique en partie par le report de l'activité en fin de cursus.

Concernant l'utilisation d'*outils* d'aide au développement collaboratif, si leur nécessité a clairement été ressentie et exprimée à l'issue de la phase 1, et si l'enseignement correspondant a été très *bien perçu*, nous constatons qu'il n'y a eu aucune utilisation effective lors des deux dernières phases. L'organisation des équipes avait changé et les délais (des dernières phases) ont sans doute été jugés incompatibles avec une mise en oeuvre effective de tels outils ; comme pour l'atelier logiciel, les étudiants s'en sortent en fait par un *travail synchrone* sur des postes de travail voisins. Nos exigences n'étaient peut-être pas suffisantes sur ce point.

E. *Encadrement*

Pour ce premier essai, le fil rouge n'a pas conduit à une notation des équipes. Toutefois, les étudiants sont très demandeurs de *feed-back* voire de validations de la part des encadrants, que ce soit en terme de qualité des exposés ou documents que des productions et choix conceptuels et techniques. Les encadrants proposés aux équipes ont souvent été perçus comme des guides/conseillers voire parfois comme des clients. Ils ont malheureusement été *peu contactés* en dehors des réunions de présentation, hor-

mis en situation de sauvetage. En ce sens, il conviendrait d'*améliorer le suivi des apprentissages* et de tendre vers une *démarche tutorale* [7], [8] en définissant plus clairement le rôle de l'encadrant à même de *renforcer le suivi* et la *démarche réflexive*.

VI. PREMIER BILAN

L'activité en était à son premier essai et de ce fait a dû supporter quelques déconvenues. Voici quelques pistes d'amélioration que nous avons identifiées.

Le produit demandé aux étudiants et sa formulation n'étaient pas de taille raisonnable quand au temps imparti à l'activité. Bien qu'il soit nécessaire de confronter les équipes à des contraintes de délais/faisabilité, nous avions largement sous-estimé l'ampleur de la tâche au regard des compétences hétérogènes des étudiants. Sur ce point, il convient donc de limiter les parties de développement pour un produit d'envergure raisonnable, par exemple en fournissant des éléments « tout faits ». De plus, il semble souhaitable d'instaurer une *phase 0* spécifiquement dédiée à la découverte et maturation du problème et des documents fournis par les encadrants (p.ex. en association avec des enseignements du tronc commun).

Les étudiants, au départ en tous cas, n'ont pas cherché à produire un système aussi simple que possible et se sont vite fourvoyés dans l'ajout de fonctionnalité, parfois exotiques. Les étudiants se laissent donc facilement emporter, ce qui conduit à complexifier encore les développement. Même si ces écueils s'avèrent parfois très formateurs, il nous semble souhaitable d'inciter, au plus tôt, à *hiérarchiser* ces ajouts et à les optionnaliser afin d'anticiper les limites de délais.

Il semble par ailleurs important de synchroniser plus finement les enseignements du mastère avec le fil rouge afin de mieux guider « à chaud » le processus de production/maintenance et justifier les propositions.

En parallèle au fil rouge, les étudiants sont accaparés par de nombreuses autres activités (p.ex. enseignements de tronc commun, recherche de stage, entretiens), et il convient de mieux en tenir compte.

VII. CONCLUSIONS

De prime abord, le fil rouge est une activité de projet par équipe visant à illustrer - *a priori* ou *a posteriori* - les différents concepts introduits dans le cursus de formation. Il guide l'étudiant dans la maîtrise de la complexité des logiciels eux-mêmes ainsi que dans la maîtrise de la complexité de leur développement en équipe. L'innovation proposée réside dans son découpage en phases successives. Celle-ci favorise l'assimilation par la pratique des différents concepts abordés, apporte aux étudiants une vision globale du cycle de vie du logiciel, et, comme nous avons d'ores et déjà pu le constater, catalyse de ce fait l'apprentissage des bonnes pratiques d'ingénierie logicielle.

La souplesse de cette activité et l'enthousiasme qu'elle suscite chez les étudiants nous confortent dans notre action. Le fil rouge constitue, d'un point de vue pédagogique, un principe d'activité projet pertinent bien que sa mise en œuvre soit délicate en terme de volume horaire et de cohabitation avec les autres activités d'enseignement. Après avoir été testé en 2002-2003, le fil rouge prendra une place centrale dans le cursus du mastère ISIC en 2003-2004. Nous veillerons particulièrement à renforcer la prise en compte des phases aval du cycle de vie logiciel.

REMERCIEMENTS

Les auteurs tiennent à remercier les étudiants du mastère ISIC 2002/2003 de l'ENST Bretagne pour leur participation active dans l'activité et pour leurs discussions informelles afin d'en améliorer la qualité. Le fil rouge, en plus des auteurs de cet article, n'aurait pu être réalisé sans l'investissement en organisation et encadrement de Antoine Beugnard, Fabien Dagnat et Robert Ogor, également enseignants-chercheurs à l'ENST Bretagne. Qu'ils en soient ici remerciés.

Références

[1] E. Aguirre, C. Jacqmot, E. Milgrom, B. Raucent, A. Soucisse, Ch. Trullemans, and C. Vander Borght. Devenir ingénieur par apprentissage actif. *Actes du 1er colloque Pédagogie par Projet dans l'enseignement supérieur : enjeux et perspectives, ENST Bretagne, Brest - France*, pages 115–122, juin 2001.

[2] Éric Cousin. Formation au projet par le projet : l'atelier logiciel. *Actes du 1er colloque Pédagogie par Projet dans l'enseignement supérieur : enjeux et perspectives, ENST Bretagne, Brest - France*, pages 169–173, juin 2001.

[3] Éric Cousin and Gérald Ouvradou. Valorisation de projets d'étudiants sous forme de logiciels libres. *Actes du 1er colloque Pédagogie par Projet dans l'enseignement supérieur : enjeux et perspectives, ENST Bretagne, Brest - France*, pages 29–36, juin 2001.

[4] Éric Cousin, Gérald Ouvradou, Pascal Pucci, and Samuel Tardieu. Picolibre : a free collaborative platform to improve students' skills in software engineering. In *2002 IEEE International Conference on Systems, Man and Cybernetics*, volume 1, page 6, Yasmine Hammamet Tunisia, October 6-9 2002.

[5] M.-C. Gaudel, B. Marre, F. Schlienger, and G. Bernot. Précis de génie logiciel. *Collection Enseignement de l'Informatique. Masson, 142 pages*, 1996.

[6] A. van Deursen, J.M. Favre, R. Koschke, and J. Rilling. *Experiences in Teaching Software Evolution and Program Comprehension. Working session at the International Workshop on Program Comprehension, IWPC 2003, Portland, Oregon - USA*, mai 2003.

[7] Caroline Verzat and Rémi Bachelet. Comment promouvoir l'autonomie en école d'ingénieurs ? le cas de l'activité professionnalisante à l'école centrale de lille. *Actes du 1er colloque Pédagogie par Projet dans l'enseignement supérieur : enjeux et perspectives, ENST Bretagne, Brest - France*, pages 55–60, juin 2001.

[8] Caroline Verzat and Rémi Bachelet. Former des tuteurs accompagnant la prise d'autonomie en école d'ingénieurs. *2nd conférence internationale IEEE SMC'02 : Systems, Man and Cybernetics, Hammamet - Tunisie*, octobre 2002.