Décembre 2010

La lettre du CFTL

Décembre 2010

Comité Français des Tests Logiciels

Le mot du président

Bonne Fin d'année à tous.

Cette période de fin d'année correspond pour la plupart d'entre nous à une période de cadeaux, de longues soirées d'hivers en famille auprès du feu. Pour les organismes de formation accrédités auprès du CFTL et pour les membres du CFTL, cette époque correspond à une surcharge de travail importante, due aux actions suivantes qui se télescopent :

1. Mise en application du nouveau syllabus Fondation du CFTL-ISTQB®

Nous vous avions informés, lors d'une lettre précédente, de la publication par l'ISTQB du nouveau syllabus niveau Fondation. Le CFTL a décidé d'appliquer cette nouvelle version à partir du 1er Octobre 2010 (version anglaise) et à partir du 1er Janvier 2011 pour la version française. Ceci implique que l'équipe en charge des accréditation du CFTL est surchargée de travail, afin de s'assurer que la qualité des formations soit au niveau que vous êtes en droit d'attendre. Les certifications obtenues avec l'ancienne version du syllabus sont toujours valables, sans limite de durée.

2. Certifications sur la Gestion des Exigences (REQB®)

La gestion des exigences est un autre point important en cours de développement au sein du CFTL. Le syllabus REQB® en Français devrait être disponible avant la fin de cette année, permettant aux organismes de formation qui le souhaitent de concevoir des formations en français dans des délais assez brefs. Des certifications, reconnues internationalement, sont déjà disponibles.

3. Référentiel des métiers et emplois de l'activité de test

L'ISTQB propose une filière simplifiée des métiers du test, mais de très nombreux métiers et emplois existent dans notre industrie. Comme le CFTL regroupe de beaucoup d'entreprises impliquées dans le métier du test, il nous a semblé important d'apporter notre contribution à la cartographie des métiers du test. Cette initiative implique à la fois des spécialistes du test, des responsables de DSI et des responsables de ressources humaines.

4. La Journée Française des Tests Logiciels du 5 Avril 2011

Les activités de préparation de la prochaine Journée Française des Tests Logiciels du Mardi 5 Avril 2011 sont bien entamées. De très nombreuses propositions de présentation nous sont parvenues, ce qui augure d'une JFTL 2011 de qualité. Il sera difficile pour l'équipe du CFTL de sélectionner les présentations retenues, tellement celles-ci sont de qualité. De nombreuses sociétés ont souhaité nous accorder leur aide et leur sponsorisation. Remercions ici nos sponsors Prestige: Microsoft, HP et Open, ainsi que tous autres sponsors qui nous font confiance. Grâce à eux et à votre présence, cette Journée Française des Tests Logiciels sera un succès.

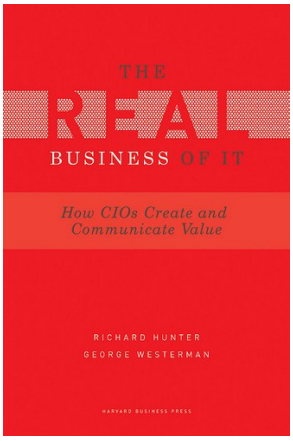
Comme vous le remarquez, l'actualité de cette fin d'année est importante et vos équipe du CFTL ont du pain sur la planche. Je souhaiterais les remercier de leur implication et vous souhaite une bonne lecture de cette lettre trimestrielle du CFTL.

Bernard Homès
Président CFTL

Dans ce numéro :

Le Mot du Président	1
Dans la bibliothèque du CFTL	2
Dates des futurs examens	3
Autours de nous	
Test-Driven Development: un pacte diabolique ?	4-7
Le coin des Organismes Accrédités	7
Les métiers du Test, appel à contribution	8
Un nouveau site pour la Journée Française des Tests Logiciels	9

Dans la Bibliothèque du CFTL



Dans cette rubrique, nous vous informons des dernières parutions d'ouvrages qui peuvent intéresser les testeurs, ou leur servir d'ouvrage de référence.

Ce mois-ci nous vous proposons le livre de **Richard Hunter et George Westerman** sur ce qu'est la réelle manière d'ajouter de la valeur par les NTIC, publié par Harvard Business Press :

The Real Business of IT : How CIOs Create and Communicate Value

La qualité et les tests de logiciels s'inscrivent dans le cadre des systèmes d'informations, et quoi de plus intéressant que de voir comment les CIO (Directeurs des SI) peuvent créer et communiquer la valeur que leurs service fournit.

Lors de la lecture de cet ouvrage, la facilité de lecture et la pertinence des arguments présentés séduisent. Qui plus est, les leçons et propositions applicables pour les systèmes d'informations sont aussi applicables pour les tests de logiciels.

Au travers de 10 chapitres, les auteurs nous guident, au travers d'exemples et de témoignages, vers une manière d'améliorer les relations entre les services informatiques et les autres services de l'entreprise. Pour nous testeurs, cela nous permet de nous remettre à notre juste place et d'identifier clairement quels sont nos réels clients et la vraie valeur ajoutée que nous sommes en mesure d'amener.

Après avoir identifié les pièges de certaines mesures, des solutions sont proposées pour créer un cercle vertueux d'améliorations, de création de nouvelles valeurs ajoutées et de transparence des investissements, lesquelles solutions peuvent relativement facilement être transposées pour les équipes de test et de développement.

« Les enseignements de cet ouvrage peuvent être extrapolés pour le métier de testeur de logiciels, et permettent de mieux comprendre les préoccupations principales de nos interlocuteurs »

Les derniers chapitres de l'ouvrage nous montrent comment intégrer les évolutions de technologies et d'organisation, de façon à augmenter la valeur ajoutée des systèmes d'information, puis à mesurer exactement la valeur ajoutée fournie à l'entreprise. L'extrapolation vers le monde du test est tellement évidente que cet ouvrage parle directement aux sentiments de la majorité des chefs de projets de test, en plus — bien sûr — des DSI à qui cet ouvrage est initialement destiné.

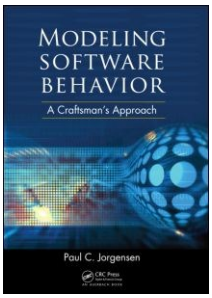
B. Homès, conférencier, éditeur et auteur du Syllabus Avancé pour l'ISTQB déclare : « *Les enseignements de cet ouvrage peuvent s'appliquer aux tests de logiciels et permettre d'avoir une meilleure reconnaissance de la profession de testeur, et de la valeur ajoutée que les testeurs apportent. Bien que se focalisant surtout pour les DSI, les leçons proposées seront utiles à toutes les personnes de notre profession. Je vous conseille vivement cet ouvrage.*».

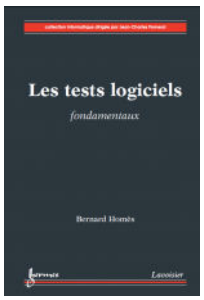
Bibliographie :

ISBN: 978-1-4221-4761-0

Editeur : Harvard Business Press

Dans une prochaine lettre trimestrielle, nous commenterons l'un des ouvrages suivants :





La lettre du CFTL
Décembre 2010

Dates des prochains examens Fondation et Avancé

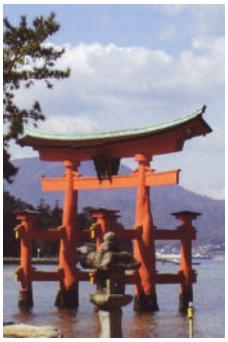
Comment s'inscrire à un examen ?
Les dates d'examen sont publiées sur le site du CFTL (www.cftl.fr rubrique Examens). L'inscription à un examen est simple, il suffit de cliquer sur le lien [détails] et de renseigner les champs requis.

Vous avez également la possibilité de contacter l'un des organismes de formations accrédités qui organisera la formation et la certification.

Pour les examens niveau Avancé, il suffit de sélectionner la date de l'examen niveau Fondation et de demander à passer un examen niveau Avancé en précisant «Gestionnaire de Tests», «Analyste de Tests» ou «Analyste Technique de Tests».

Date	Titre	Emplacement	Coût
28.01.2011	Examen niveau Fondation CFTL/ISTQB	Paris	€ 250
22.02.2011	Examen niveau Fondation CFTL/ISTQB	Nantes	€ 250
04.03.2011	Examen niveau Fondation CFTL/ISTQB	Paris	€ 250
01.04.2011	Examen niveau Fondation CFTL/ISTQB	Lyon	€ 250
05.04.2011	Examen niveau Fondation CFTL/ISTQB	Paris	€ 250
20.04.2011	Examen niveau Fondation CFTL/ISTQB	Marseille	€ 250
19.05.2011	Examen niveau Fondation CFTL/ISTQB	Bordeaux	€ 250

Autour de nous



Conférence ISTQB à Tokyo

Le Comité Japonais des Tests Logiciels (JSTQB), a accueilli la dernière réunion de l'ISTQB à Tokyo (Japon) courant Octobre 2010. Cette réunion a été l'occasion pour les membres de l'ISTQB (dont le CFTL) d'accueillir le Portugal en son sein.

Les prochaines conférences de l'ISTQB auront lieu à Prague (République Tchèque) en Mars 2011 et à Rome (Italie) en Juin 2011 et à Shangaï (République Populaire de Chine) en Novembre 2011.

L'école des Mines d'Alès choisi le CFTL

De nombreux membres du CFTL sont impliqués dans le transfert de connaissances vers les universités, et les grandes écoles sont preneuses de cette implication. L'implication de l'école des Mines d'Alès va un peu plus loin car la décision a été prise de proposer à leurs étudiants le cours Knowledge-Department version 2010, accrédité par le CFTL (module Fondation et partie du module Avancés) suivi de la certification par le CFTL. Cette initiative est la première en France, espérons qu'elle soit suivie de nombreuses autres universités et grandes écoles.

Le CFTL en Tunisie

Le Président du CFTL a été sollicité par des organismes Tunisiens pour fournir du conseil et de l'assistance dans l'exécution de sessions d'examen d'accréditation à destination des testeurs Tunisiens.

Le CFTL en Corée

Le CFTL, représenté par son Trésorier, a participé aux conférences SSTC'2010 (Seoul Software Testing Conference) en Corée du Sud. Ceci a été l'occasion de resserrer les liens étroits entre les comités Coréens et Français.

On parle du CFTL dans la presse

De nombreux organes de presse, dont 01 informatique, DSIsionnel, IT-Expert, Best Practices et Qualité Référence, considérant que les tests de logiciels sont importants dans le monde d'aujourd'hui ont décidé de sponsoriser la Journée Française des Tests Logiciels 2011. Le CFTL est aussi cité dans la presse comme la référence du test logiciel en France, en reprenant les avis d'experts proposés par le CFTL et développant l'intérêt des certifications proposées par le CFTL.

Venez visiter notre Carré Presse au : http://www.carrepresse.biz/cpresse/carre.php

La lettre du CFTL
Décembre 2010

Test-Driven Development: un pacte diabolique?

Introduction

Les Développements dirigés par les tests (TDD Test-Driven Development) sont récemment apparus comme le modèle type de développement de logiciels du modèle Agile. Dans sa forme la plus pure, le développement de logiciels selon les préceptes TDD consiste uniquement en:

- Un ensemble de « user stories» (histoire utilisateur),
- Un ensemble de tâches associées à chaque user story,
- Un ensemble de cas de tests pour chaque tâche, et du code source conçu, avec des variables et des unités bien nommées, qui ont été refactorisées.

Un Pacte Diabolique se réfère à la légende de Faust datant de l'époque médiévale et de la renaissance, dans laquelle un savant, (Faust) négocie un accord avec le diable (Méphistophélès) dans lequel il échange son âme pour une extension de ses pouvoirs terrestres. Dans la version de Christopher Marlowe, le Dr. Faust a des pouvoirs suprêmes pendant 24 années, et fini coupé en morceaux sur un tas de fumier. La légende a parfois une fin heureuse où le diable est battu, comme dans le classique de la musique country (américaine) appelé "The Devil in Georgia" (ou le duel au violon) où le diable parie son violon d'or qu'il joue mieux au violon que le héro, Johnny (Johnny gagne). Dans un usage contemporain, un pacte diabolique réfère habituellement sur un choix opportuniste plutôt qu'excellent, ou une préférence de récompenses immédiates par opposition à des bénéfices plus durables.

Le processus TDD idéal

Dans sa forme la plus pure, TDD commence avec un squelette de programme, soit une procédure en langage traditionnel, ou une méthode en langage objet. Le développeur en TDD conçoit un ensemble de cas de test pour la première tâche de la première «user story», et exécute ces cas de test sur ce code (vide). Bien sûr, tous les tests échouent. Ensuite, le développeur écrit juste assez de code pour que les cas de test passent tous. Ceci requiert de la discipline, car il y a une tendance naturelle à écrire plus de code que strictement nécessaire. L'ensemble des cas de test sont exécutés à nouveau sur le nouveau code. Si tous les tests passent, le code est considéré comme solide, et un nouvel ensemble de cas de tests est généré (pour la tâche suivante). Le code existant est exécuté avec les cas de test étendus, et, comme on s'y attend, échoue pour les nouveaux cas de tests. On ajoute ensuite juste assez de code de façon à passer avec succès les nouveaux cas de tests, et les cas de tests sont répétés. Si un cas de test échoue, la cause doit se trouver dans le code ajouté récemment. Le développeur TDD vérifie le code, en utilisant le(s) test(s) défaillant(s), corrige le défaut et réexécute la suite de cas de tests. Ce cycle continue jusqu'à ce que toutes les tâches de toutes les «user stories» soient testées et implémentées.

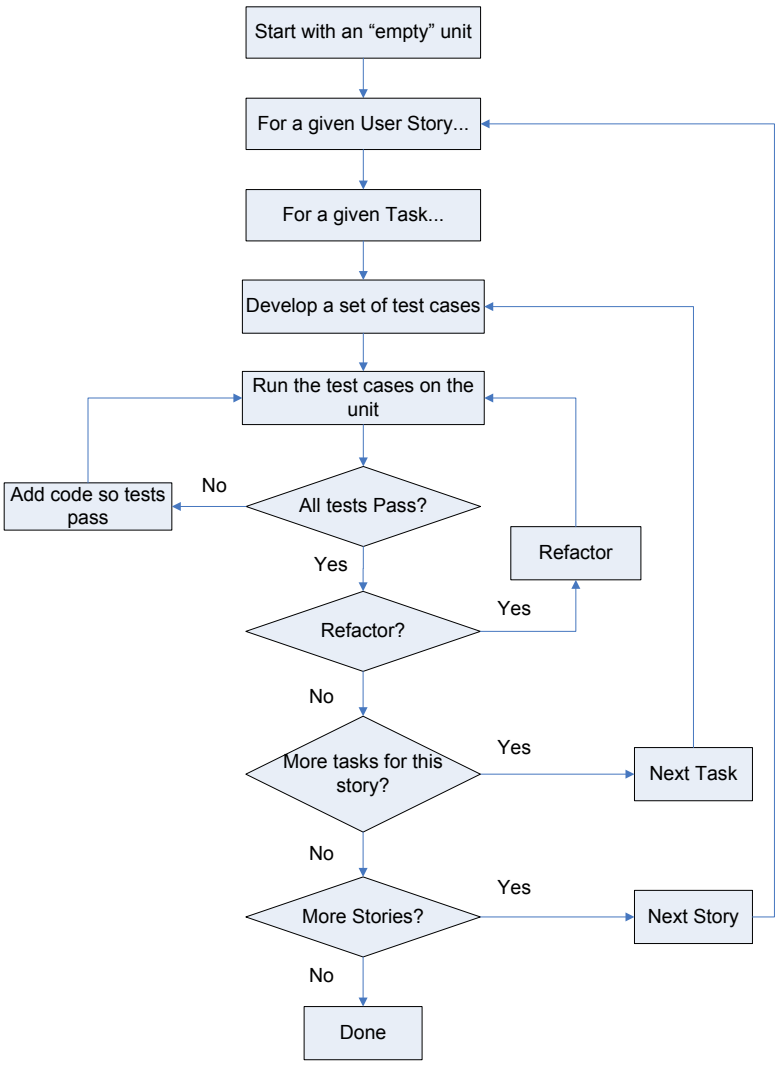
A divers points dans le processus TDD, le développeur peut s'arrêter afin de réécrire (refactoriser est le terme agile) le code existant dans une forme plus élégante. Ceci peut impliquer l'identification de nouveaux composants (procédures, méthodes), ou cela peut être une simple restructuration du code. Une longue série d'instructions Si-Alors (If-Then), par exemple, peut être refactorisée en une instruction Case/Switch, ou une logique de If-Then-Else imbriqués en profondeur peut être refactorisée en une logique composée de conditions complexes. Ce processus TDD complet est décrit dans la Figure 1.

Les Avantages du Processus TDD Idéal

L'avantage principal du processus TDD idéal est l'isolation des defaults qui est quasi garanti par les cycles courts Test-Code-Test. Quand les tâches correspondant à une user story sont petites, le nouveau code est alors créé en de petits incréments. Le test isole les defaults dans le nouveau code de façon quasi immédiate.

Un second avantage, tout aussi important, est que « il y a toujours quelque-chose qui fonctionne » (le mantra agile). Comme tout le processus TDD est dirigé par des «user stories» qui reflètent les priorités des utilisateurs, à la fin d'un cycle complet pour une "user story" (voir Figure 1), le développeur TDD peut livrer une fonctionnalité testée au client, dans l'ordre qui a été

Test-Driven Development: un pacte diabolique? (suite)



demandé. Pris ensemble, une isolation efficace des défauts et un logiciel qui fonctionne continuellement, donnent un niveau de confiance important à la fois au client et au développeur. Ceci doit être mis en opposition avec la relation parfois conflictuelle entre clients et développeurs dans un cycle de développement séquentiel (en cascade ou en V).

Le troisième avantage majeur revendiqué par ceux qui préconisent TDD est l'accroissement de productivité qui résulte de la non exécution des autres étapes d'un mode de développement non-agile : inspections techniques, conception générale et détaillée, niveaux de plans de tests détaillés, et documentation, contribuant ainsi à une productivité apparemment accrue. ("apparemment " annonce ici le pacte diabolique.)

Le quatrième avantage, indéniable, est que TDD est amusant. Les participants ont des succès rapides et fréquents, et sont rarement englués dans des aspects de taille et de complexité.

Questions sur le Processus TDD idéal

La communauté agile se réfère souvent aux rapports déprimants du GAO américain (US Government Account Office) et du groupe Gartner sur les faible taux de succès des logiciels développés avec la méthode en Cascade. Ces rapports ont récemment été discrédités, donc les arguments de la communauté agile se résument à faire un épouvantail des projets basés sur la méthode de développement en cascade. La majorité des projets non-agiles actuels utilisent une sorte de cycle itératif.

1. Comment les projets TDD montent-ils en puissance pour des applications de grande taille? Pouvons-nous imaginer quelque-chose comme un système avionique embarqué, ou un commutateur téléphonique développé avec la méthode TDD ? La réponse de TDD à ceci est que la seule limite en taille est la séquence des « user stories » qui dirigent le processus. Bien sûr, il y a les inévitables limitations en terme de délais et de budget, mais, tant que le client ajoute des « user-stories », le processus continue. Un problème avec ceci est que la séquence des « user stories » est plate, il n'y a pas de notion de hiérarchie. Les approches non-agiles traitent la taille par l'utilisation de modèles, de niveaux d'abstraction, et une certaine notion de hiérarchie. Rien de tout cela n'est présent dans TDD. Un autre problème est qu'il y a des limites à ce qu'un développeur agile peut « garder à l'esprit » à un moment donné.

2. Comment TDD traite-t-il la complexité applicative ?

Une manière de décrire la complexité qui s'applique à la fois aux projets TDD et aux projets non-agiles est de créer une matrice d'incidence qui relie les éléments comportementaux avec les éléments structurels : ceci pourrait être les caractéristiques par rapport aux procédures, ou les cas d'utilisation par rapport aux classes. Ceci peut clairement être effectué à la fois dans les projets non-agiles et les projets TDD. Les données introduites dans la matrice montrent quels éléments de structure sont requis pour supporter l'exécution de l'élément comportemental, *càd,* quelle classe supporte un cas d'utilisation donné. Dans ce cadre, que la matrice soit dense ou éparse est un indicateur utile de la complexité. Une matrice d'incidence dense veut dire que les composants structurels sont liés de multiples façon – indiquant ainsi que les interfaces seront un souci. Dans les projets TDD, cette matrice est généralement éparse, alors que dans des projets de grande taille et des projets complexes, elle est

Test-Driven Development: un pacte diabolique? (suite)

dense. Il n'y a pas grand chose dans le processus TDD qui aide à traiter la complexité des interfaces. L'espoir principal est la refactorisation, et ceci devient à la fois de plus en plus fréquent et de plus en plus difficile au fur et à mesure de l'accroissement de la complexité.

3. Comment TDD traite-t-il la compréhension du système ?

Les seuls éléments d'un projet TDD qui supportent la compréhension du système sont des éléments de code bien nommés, des « user stories », et peut-être, une liaison entre les « user-stories » et le code. (Cette liaison n'est pas requise dans les projets purement TDD.) D'un autre coté, les projets non-agiles ont des modèles avec différents niveaux d'abstraction. Les éléments de conception [(documentation, architecture, etc.)], en plus d'un code correctement commenté, sont une aide à la compréhension d'un programme, et ni l'un ni l'autre ne sont présents dans des projets TDD. De plus, la compréhension du système devient de plus en plus difficile au fur et à mesure de l'accroissement de la taille du système.

4. Y a-t-il une quelconque garantie que les cas de tests développés dans le processus TDD constituent réellement un bon test ?

La raison principale pour laquelle nous testons est que tout le monde peut se tromper. Dans des projets non-agiles, il y a plusieurs niveaux d'inspections techniques et plusieurs niveaux de tests. Dans un projet TDD, les « user stories », suivies des cas de tests, conduisent l'effort de codage. Certes, le code passe les tests avec succès, mais il n'y a aucune vérification que les cas de tests TDD sont suffisants. On pourrait argumenter que les projets TDD utilisent des tests basés sur le code, mais des praticiens sérieux de TDD acceptant que pour être efficaces, ils ont besoin d'une compétence bien établie en théorie et en techniques de test. L'absence de toute modélisation suggère que TDD sera efficace au niveau unitaire [(tests de composants)], et ne testera que des cas d'utilisation (et des « user stories ») au niveau système. Il y a peu d'opportunités de tests d'interfaces (tests d'intégration), et aucune opportunité de tester en se basant sur les modèles [MBT Model Based Testing]. Des niveaux de tests supplémentaires, tels du test de stress et de sécurité, ne seront exécutés que s'ils sont "requis" par les séquences de « user stories ».

5. Comment TDD traite-t-il les exigences non-fonctionnelles, telles que la performance, la fiabilité, la sécurité, la capacité de débit ou la bande passante, et la maintenabilité ?

Rien dans TDD ne permet de traiter ces questions. Les projets non-agiles ont traité avec succès de ces questions, principalement avec une modélisation rigoureuse et de l'analyse. Des niveaux de tests plus profonds tels du test de stress ou de sécurité sont uniquement exécutés s'ils sont « requis » par les séquences de « user stories ».

6. Vu que les cas de tests au niveau des tâches dirigent le processus TDD, que se passe-t-il si ces tests sont inadéquats, p.ex. incomplets, insuffisants ou incorrects ?

Les cas de tests TDD génèrent une sorte de vision tunnel pour le développeur TDD – Si les tests passent avec succès, le code est correct, et c'est tout. Il n'y a pas de vérifications croisées similaires à celles générées par l'utilisation responsable de tests basés sur les spécifications et sur le code.

7. Considérons les types de défauts qui ne peuvent être détectés que par le test du flux de données (technique basées sur les points dans un programme où les variables sont définies et les points où ces valeurs sont utilisées). L'hypothèse TDD est que le passage avec succès de tous les tests de toutes les tâches signifie que le code est correct. Dans une longue séquence de "user stories" et de tâches, comment TDD garantit-il que les cas de test au niveau tâche identifieront les défauts du flux de données dans un code qui correspond à des « user stories » qui ne sont pas en séquence dans la demande de l'utilisateur?

Il n'y a rien dans le processus TDD qui traite explicitement les défauts des flux de données. Cette déficience s'accroit inévitablement en proportion avec la taille du projet.

8. Par rapport aux défauts des flux de données, considérons un projet TDD dans lequel plusieurs paires de programmeurs (un autre précepte agile) travaillent en parallèle, probablement sur des « user stories » séparées. Quel est le processus TDD qui garanti que les composants créés et testés isolément avec succès seront sujet à une forme de test d'intégration ?

Le seul espoir dans ce cas est un très bon test de niveau système. TDD n'a pas le concept traditionnel de test d'intégration autre que de réexécuter complètement tout le code existant. A nouveau, ce problème est proportionnel à la taille.

Test-Driven Development: un pacte diabolique? (suite)

9. Comment la documentation délibérément parcellaire, supporte-t-elle le besoin éventuel de maintenance du programme ?

Ceci est un problème sévère. La seule réponse de la communauté TDD est que le code devrait être écrit clairement avec des conventions de nommage utiles.

10. Comment la refactorisation répétée, un processus exclusivement de bas en haut, produit-il une conception élégante ?

Ceci est une affirmation très controversée, supportée par bien peu de preuves. Une refactorisation répétée, si elle est faite efficacement, résultera en du code élégant, mais une conception élégante est un niveau d'abstraction au dessus de celui du code source, et TDD évite de façon explicite une phase de conception séparée. Dans des expériences faites avec des étudiants à Grand Valley State University, une conception « élégante » n'a que très rarement été observée sur des projets de type TDD.

11. Dans des développements de logiciels traditionnels (non agiles), la phase de conception est le point où les composants du logiciels sont identifiés / créés, et ensuite améliorés par rapport aux considérations de performance (voir point 5 ci-dessus). Comment une « vision tunnel » générée par TDD peut-elle amener une conception qui traite efficacement des exigences de performances ?

Une fois de plus, rien dans TDD ne traite directement des aspects de performances. Une possibilité serait que ce soit pris en compte lors de la refactorisation.

12. Dans une longue séquence de « user stories », comment TDD peut-il détecter des inconsistances même légères entre les tâches d'une « user story », ou entre les « user stories » elles même ? Il n'y a rien dans les processus TDD qui ne traite la détection d'inconsistances subtiles.

Un pacte diabolique ?

Ces douze questions peuvent toutes être adressées par les approches de développement logiciel non-agiles. Certes, un processus de développement mature, discipliné est requis, mais cela peut se faire et cela se fait. Dans la mesure où TDD ne peut pas répondre à ces questions, et qu'il y a probablement d'autres questions aussi sévères qui ne sont pas traitées, TDD est un pacte diabolique. Il échange un temps de réponse court et élimine les efforts traditionnels pour l'assurance de meilleurs réponses à ces douze questions.

Le professeur Paul C. Jorgensen, Ph.D. fait partie du comité d'écriture du syllabus ISTQB niveau Avancé et a été récemment élu vice-chair en charge du glossaire de l'ISTQB.

Paul C. Jorgensen, Ph.D., Professor School of Computing and Information Systems, Grand Valley State University, Allendale, MI 49341, USA—Traduction: Bernard Homès

Le coin des Organismes Accrédités

La plupart des organismes de formation accrédités ont soumis leurs supports de formation mis à jour au CFTL pour re accréditation. Les examens de certification en français s'appliqueront à la nouvelle version du syllabus à partir du 01/01/2011.
Le CFTL tient à féliciter les organismes de formation accrédités pour leur réactivité et les membres de l'équipe d'accréditation pour le travail effectué.

Le schéma REQB d'Ingénierie des Exigences

Avec de nombreuses sociétés intéressées, le schéma de certification sur l'Ingénierie des exigences REQB prend son essor. Le syllabus est traduit et un glossaire est constitué, ce qui permettra aux organismes de formation et aux candidats de préparer les formations et les examens de certification. Les modalités financières et d'accréditation sont disponibles sur simple demande auprès du CFTL.

Merci à Alain Ribault de Kereval pour l'organisation de la traduction

Appel à contribution pour définir les métiers du test en France

Ce n'est un secret pour personne, les activités de test sont de plus en plus visibles dans les projets informatiques. Tout le monde convient de l'impérieuse nécessité de les formaliser, de les organiser et de les professionnaliser. Selon les analystes, cette activité représente un investissement dans les projets pouvant aller jusqu'à 30% de la charge globale. C'est dire l'importance qu'il faut y accorder, ne serait-ce que sur le plan budgétaire. Mais c'est également une garantie de bonne terminaison des projets, petits ou grands. Il n'est pas utile de rappeler ici que de la qualité des tests dépend la qualité globale du logiciel et surtout de sa bonne adéquation avec le besoin initial.

Il existe aujourd'hui tout un arsenal de méthodes et d'outils spécifiques à cette activité. Les technologies de l'information et de la communication ont envahi notre quotidien, partout où il y a de l'électronique, des systèmes, des réseaux, partout il y a des logiciels dont le nombre et la complexité ne cesse d'augmenter.

Tester est devenu une affaire de spécialistes, de professionnels. Le professionnalisme de ces « testeurs » passe indiscutablement par des cursus de formation pour développer les compétences conduisant à une certification internationalement reconnue. Il reste cependant encore beaucoup à faire pour valoriser cette activité, en commençant par décrire les métiers qui la constitue. Qui n'a pas utilisé ce qualificatif de « testeur » avec un soupçon de condescendance. Cela rappelle ce que l'on disait des informaticiens il y a de nombreuses années, ils étaient soit programmeurs soit exploitants. Que de métiers sont apparus depuis un demi-siècle. Les TIC sont devenues l'un des secteurs qui participent au 2e ou 3e rang du PIB et crée le plus d'emplois (25% de l'emploi cadre tous secteurs confondus, source APEC).

Il est grand temps que le CFTL s'approprie la responsabilité de valoriser ces métiers, en commençant par les décrire en s'appuyant notamment sur des expériences qui ont vu le jour ici ou là. Nous avons décidé de lancer une initiative pour élaborer **un référentiel des métiers et des emplois de l'activité de test**.

Nous définirons la carte du développement de ces métiers, décrirons les activités, les compétences associées à chacun d'entre eux ainsi que les passerelles vers d'autres activités. Un référentiel, c'est la clef de voute de la politique RH. Ce sera aussi un outil de communication pour assurer la promotion de ces métiers dans les écoles et universités, auprès des entreprises et des professionnels du secteur.

Si vous souhaitez vous impliquer dans ces travaux et apporter votre contribution, n'hésitez pas à contacter le CFTL info@cftl.net pour faire acte de candidature avec le bulletin de participation ci-contre.

Notre prochaine conférence téléphonique est prévue le 16/12/10 à 14h30. Nous avons prévu de nous réunir par téléphone au moins une fois toutes les 3 à 4 semaines pour recueillir les contributions des uns et des autres et faire converger les travaux en vue d'obtenir un premier résultat pour le début d'année 2011, puis la publication lors de la journée CFTL.

Dominique DUFLO, animateur de l'atelier Référentiel Métiers du Test au sein du CFTL

Bulletin de participation

Je suis intéressé par les activités du CFTL pour mise en place d'un référentiel des métiers et des emplois de l'activité de test.

Je souhaite participer à titre individuel à la définition des métiers du test

- ☐ Comme contributeur
- ☐ Comme réviseur
- ☐ Non

Je suis intéressé par les résultats de ces activités pour les implémenter

- ☐ En tant que testeur
- ☐ En tant que DSI
- ☐ En tant que DRH

Je suis membre du CFTL

- ☐ Membre individuel
- ☐ Membre Société
- ☐ Non membre

Je souhaite participer en tant que société à la définition des métiers du test

- ☐ Comme contributeur
- ☐ Comme réviseur
- ☐ Comme utilisateur

Ma société utilise des testeurs certifiés par le CFTL

- ☐ Oui
- ☐ En cours
- ☐ Pas encore

Je souhaite des informations sur les activités du CFTL

- ☐ Sur les certifications niveau Fondation
- ☐ Sur les certifications niveau Avancé

Nom

Adresse

Téléphone

email

@

Commentaires :

Comité Français des Tests Logiciels

14 rue des Anthémis
B.P. 49
83.190 Ollioules
France

La lettre du CFTL
Décembre 2010

Journée Française des Tests Logiciels

Un site internet spécifique pour la JFTL

Conscient que l'événement majeur de l'année, la Journée Française des Tests Logiciels n'était pas suffisamment mis en avant sur son site, le CFTL a décidé de vous proposer un site spécifique pour cet événement :

http://www.jftl.org

Ce site reprendra les informations présentes sur le site du CFTL et se focalisera sur le prochain événement. Comme l'inscription préalable à cette journée est obligatoire, vous pouvez utiliser ce site pour vous pré-inscrire à l'événement via ce site.

Présentation d'Eric Riou du Cosquer à SSTC'2010 à Seoul.

Des conférences dédiées au test logiciel, régulièrement organisées dans le monde entier, sont une occasion formidable de rencontrer des experts en test et des industriels d'autres pays et de discuter avec eux.

J'ai eu l'honneur de participer à la conférence SSTC (voir aussi Autour de Nous page 3) pour y parler du lien fort qui doit exister entre le Plan de Test et l'outil de gestion des tests. En posant la question "Comment corréler le plan de test et l'outil de gestion des tests?" j'ai partagé mon expérience sur le sujet en donnant des exemples et des conseils sur la façon d'associer les deux. Il est en général très bénéfique d'avoir sur tout projet un plan de test et un outil de gestion des tests, l'un ne va pas sans l'autre. Mais il est aussi important, premièrement, d'utiliser le contenu du plan de test pour construire le référentiel de test, par exemple sa structure, et, deuxièmement, d'utiliser l'outil de gestion des tests pour fournir automatiquement certains livrables présentés dans le plan de test ou encore pour mesurer automatiquement certains critères introduits dans le plan de test. Un processus de construction simultanée du plan de test et du référentiel de test outillé a aussi été présenté. De nombreuses et bonnes questions à l'issue de la conférence ont confirmé que même si le sujet peut sembler basique, il est important de bien y travailler avant de se lancer dans des techniques plus avancées en matière de test.

Eric Riou du Cosquer, Trésorier CFTL

Comité Français des Tests Logiciels

Association CFTL
BP.49
83190 Ollioules
France

Messagerie : info@cftl.fr

Le CFTL décline toute responsabilité concernant la mise en place des techniques proposées dans cette lettre. En effet, suivant les contextes, ces dernières peuvent ou non, être recommandées.
Les avis & opinions proposées sont ceux des auteurs des articles et ne représentent pas spécifiquement l'opinion du CFTL ou de l'ISTQB.

Directeur de publication: Bernard Homès, CFTL asso., B.P. 49, 83191 Ollioules

www.cftl.fr

Contribuer à cette lettre d'information...

Vous pouvez contribuer à cette lettre d'information en nous soumettant des articles. Ils seront revus par des experts du CFTL et en fonction de leur pertinence seront inclus dans une prochaine lettre.

Vous êtes tous les bienvenus pour contribuer à cette lettre d'information.

Néanmoins certaines règles sont imposées :

Pour un particulier

Envoyez-nous votre article comportant 1 page de texte maximum et au moins un schéma. N'oubliez pas de vous présenter et de joindre votre photo. Nous le publierons dans la rubrique « Technique ».

Pour une société industrielle

Vous voulez faire partager votre expérience ? Envoyez-nous votre article comportant 1 page de texte maximum et au moins un schéma. N'oubliez pas de vous présenter vous et votre société et de joindre votre photo. Nous le publierons dans la rubrique « Technique » ou « Retours d'expérience ».

Pour une SSII ou un Editeur

En échange d'une contribution financière, vous pouvez présenter en une page maximum, votre expertise ainsi que vos services dans la rubrique « La Société Test du mois ».