Software Qualität Übung 1

Regressionstests mit JUnit
Versionskontrolle mit CVS
Bugtracking mit Bugzilla

1. Informationen

1.1 Formales

Abgabetermin: Freitag 20.April 2007, 18.00 CET (Central European Time)
Das abzugebende Material ist in den Teilaufgaben erwähnt. Abgaben sind per e-Mail an cramer@ifi.unizh.ch und samuel.fricker@ifi.unizh.ch zu schicken, falls nicht anders angegeben.

Die Aufgaben sollen mit Eclipse (http://www.eclipse.org) gelöst werden. Die Computerräume am IFI haben Eclipse installiert.

Bei Fragen: e-Mail an cramer@ifi.unizh.ch oder Uniboard.
Literatur zur Arbeit mit CVS in einer Eclipse-Umgebung finden Sie zum Beispiel unter http://wiki.eclipse.org/index.php/CVS_FAQ.

1.2 Struktur der Übung

Ziel der Übung ist, den Ablauf von Regressionstests, Versionskontrolle und Bugracking unter Verwendung geeigneter Werkzeuge kennenzulernen.
Hierzu sind 2 Aufgaben zu lösen:
Teil A: JUnit Tests werden auf einer einfachen Mathematikbibliothek aufgebaut und durchgeführt.
Teil B: Eine einfache Banking-Anwendung wird in Gruppenarbeit weiterentwickelt, koordiniert durch Bugzilla und CVS.

1.3 Gruppen

Der Teil A ist in Einzelarbeit zu lösen.
Der Teil B wird in 2-er Gruppen gelöst, wobei die Gruppenzuteilung in der Übungsstunde am 2. April gemacht wird. Wer nicht an der Übungsstunde teilgenommen hat, kann sich per e-Mail melden, um einer Gruppe zugeteilt zu werden.

2. Aufgabestellung

2.1 Teil A

Ziel dieser Übung ist, Regressionstests verwenden zu können, um hohe Softwarequalität trotz ändernden Code gewährleisten zu können. Dafür werden JUnit-Tests auf einer einfachen MathematikBibliothek aufgebaut und wiederholt durchgeführt.

Vorbereitung: Laden Sie das Projekt MathematikBibliothek vom Repository (File **à** Import **à** CVS / Project from CVS). Um zu verhindern, dass Sie ihre Änderungen auf das Repository laden, trennen Sie das Projekt vom Repository (rechte Maustaste **à** Team **à** Disconnect).

Host: 130.60.157.149
Repository Path: \swq
User: SwQ
Password: SwQ

Abgabe: Ein zip-File mit den Projekten MathematikBibliothek und einem Textdokument mit Antworten zu den gestellten Fragen.

Teil A.1

Erstellen Sie eine Applikation, welche Zahlen aus einem Vektor mit der Zahl 100 multipliziert. Dafür werden sollen folgende Methoden implementiert werden.

- *boolean isNumber(Object entry).* Gibt true zurück falls entry eine Zahl ist, welche mit 100 multiplizert werden kann.
- *int multiplyByHundred(int number).* Gibt number * 100 zurück.
- *Vector getMultiplyVectorByHundred(Vector entries).* Gibt einen Vektor zurück, welcher alle Einträge von entries mit 100 multipliziert (sofern möglich). Die Einträge von Entries können beliebig sein (also zum Beispiel auch Strings).

Gehen Sie wie folgt vor:
- Schreiben Sie die Signatur der oben angegebenen Methoden
- Für jede Methode
    - Schreiben Sie JUnit-Tests nach dem Blackbox-Ansatz
    - Implementieren Sie die Methode und testen Sie diese

Beantworten Sie folgende Fragen:
In welcher Reihenfolge haben Sie die Methoden implementiert. Warum?

Teil A.2

Erstellen Sie eine Applikation, welche für Elemente eines Vektors prüft, ob es sich um eine Primzahl handelt.

Vorbereitung: Kopieren Sie dazu das gesamte Projekt und nennen Sie die Kopie MathematikBibliothek2. Kommentieren Sie die Testfälle zur Methode *multiplyVectorByHundred* aus.

Folgende Methoden sollen implementiert werden:

- *boolean isNumber(Object entry).* Gibt true zurück falls entry eine ganze, positive Zahl ist.
- *boolean getPrime(int number).* Gibt true zurück, faslls number eine Primzahl ist.
- *Vector getPrimeNumbers(Vector entries).* Gibt einen Vektor zurück, welcher alle Einträge von entries enthält, welche eine Primzahl sind. Die Elemente von entries können beliebig sein.

Gehen Sie wie folgt vor:
- Schreiben Sie die Signatur der oben angegebenen Methoden
- Für jede Methode
    - Schreiben Sie JUnit-Tests nach dem Blackbox-Ansatz
    - Implementieren Sie die Methode und testen Sie diese

Beantworten Sie folgende Fragen:
Entfernen Sie die Kommentierung der Testfälle zur Methode *multiplyVectorByHundred().* Testen Sie ihr Programm erneut. Was ist das Resultat der Tests? Wie ist das Resultat zu erklären?

Teil A.3

Erstellen Sie in Projekt MathematikBibliothek3 welches die Anforderungen beider Teile (MathematikBibliothek und MathematikBibliothek2) erfüllt. Dazu gehören auch JunitTests zu allen verwendeten Methoden.

Gehen Sie gleich vor, wie bei den Teilaufgaben A.1 und A.2.

2.2 Teil B

Ziel dieser Aufgabe ist eine einfache Banking-Anwenkdung in Gruppenarbeit weiterentwickeln, koordiniert durch die Verwendung von Regressionstests, CVS und Bugzilla.

Abgabe:
- Vollständig lauffähige Version des Projektes auf dem CVS Repository, inkl.
    - vorgegebene Aufträge
    - Aufträge von Partner
    - JUnit-Tests
- Beantwortete Fragen.

Vorbereitung:
Laden Sie vom Repsitory das Projekt mit ihrem Gruppennamen (z.B grp1 für Gruppe1) herunter. Sie müssen sich dazu im Netz der Uni befinden (z.B. über VPN).

Host: 130.60.157.149
Repository Path: \swq
User: SwQ
Password: SwQ

Öffnen Sie einen Bugzilla-Account auf http://130.60.157.149 mit der e-Mailadresse, welche Sie bei der Gruppenzuteilung angegeben haben. Auf dem Bugzilla-Server befindet sich ein Projekt (Product) mit welches Ihrer Gruppennummer entspricht (zum Beispiel grp1 für Gruppe1).

Gehen Sie wie folgt vor:
Ihre Arbeitsanweisungen finden Sie im Projekt mit Ihrem Gruppennamen in der Form von Bugzilla Bugs. Sie müssen alle Bugs bearbeiten. Falls Sie Fragen zu den Bugs haben stellen Sie diese im Bug-Record unter Additional Comments.

Wenn Sie einen Bug bearbeiten ändern Sie den Status auf ASSIGNED so dass das andere Gruppenmitglied weiss, dass dieser Bug bearbeitet wird. Ist ein Bug behoben soll der Status auf „FIXED“ gesetzt werden. Nutzen Sie

Eine Person Ihrer 2er-Gruppe ist für die Klasse Konto verantwortlich, die andere Person für die Klasse Verwaltung. Bearbeiten Sie die Bugs welche ihren Verantwortungsbereich betreffen. Sie können jedoch auch in der anderen Klasse Änderungen durchführen falls dies notwendig ist.

Für jegliche Änderung wird, gleich wie im Teil A, zuerst ein JUnit-Test geschrieben und dann implementiert. Sobald die Applikation erfolgreich alle Tests bestanden hat, kann sie eingecheckt werden. Die Version welche sich auf dem Repository befindet muss immer alle Tests bestehen! Falls sich Konflikte ergeben beheben Sie diese sofort. Denken Sie daran regelmässige Updates zu machen, so dass Sie immer die neuste Version des Projektes vor sich haben.

Wenn Sie ihre Arbeit einchecken achten Sie darauf, dass im Kommentar zum Commit ihr Name an erster Stelle steht damit nachvollzogen werden kann, wer welche Änderung gemacht hat.

Zusätzlich sollen Sie für die Klasse ihres Partners (Konto oder Verwaltung) vier weitere Verbesserungsvorschläge als Bugs eingeben. Diese müssen vor dem 11. April eingetragen sein und ebenfalls bis zum Abgabetermin bearbeitet und behoben werden.

Beantworten Sie folgende Fragen:

- Warum ist es besser, oft kleine Änderungen aufs Repository zu laden?
- Warum muss vor dem commit ein update gemacht werden?
- Ist es möglich ein File welches gelöscht wurde wieder vom Repository zu holen? Warum?
- Es gibt Repositories welche einen Locking-Mechanismus verwenden. Das heisst ein File kann nur immer von einer Person bearbeitet werden. Was sind die Auswirkungen davon? Geben Sie Vor- und Nachteile an.
- Kann CVS automatisch Konflikte lösen? Falls ja, wie? Falls nicht, inwiefern unterstützt CVS bei der Lösung von Konflikten?
- Sie möchten wissen inwiefern die ursprünglichen Tests auf der endgültigen Lösung noch laufen. Wie können Sie diese zurückholen?
- Aus Versehen löschen Sie eine Klasse. Welche Möglichkeiten haben Sie diese wenigstens teilweise wieder zurückzuholen?
- Sie haben im Software-Praktikum an einem grösseren Projekt mit mehreren Teammitgliedern gearbeitet. Haben Sie damals ein CVS verwendet?

Falls ja, beschreiben Sie die Erfahrungen welche sie damit gemacht haben. Falls nein überlegen Sie sich, wie sich die Arbeit verändert hätte, wenn Sie ein CVS gehabt hätten. Gab es spezifische Probleme welche hätten verhindert werden können?