Universität Zürich[UZH]
Institut für Informatik

s.e.a.l. software evolution & architecture lab
RERG

Software Engineering Übung 6

Softwaretests

1 Informationen

1.1 Daten

- Ausgabe Di 29.11.2011
- Abgabe So 11.12.2011 bis 23:59 Uhr
- Besprechung am Di 20.12.2011 um 11:50 Uhr

1.2 Formales

Die Lösungen sollen als PDF Datei mit dem Namen **Ex[*n*]_[*NameA_NameB_NameC*].pdf** abgegeben werden, wobei [*n*] die Nummer der Übung ist und [*NameA_NameB_NameC*] die Nachnamen der Gruppenmitglieder sind. Die PDF Datei sollte ausserdem ebenfalls Ihre Namen und Matrikelnummern beinhalten.

Mailen sie Ihre Lösungen vor dem Abgabetermin an `todoran@ifi.uzh.ch` und `nhoby@access.uzh.ch`. Der Betreff der E-mail sollte mit **[SE EX HS11]** beginnen. Falls Sie zusätzliche Abgabematerialien (z.B. Source Code) haben, mailen Sie bitte ein Archiv (.zip-File), welches alle Dateien, einschliesslich dem PDF, enthält. Benennen sie das Archiv anhand der oben erwähnten Konventionen. Als Abgabedateien sind nur PDF-Dokumente und Source Code Dateien erlaubt, keine Bilder.

Die Übungen sollen in 3er Gruppen gelöst werden. Jedes Gruppenmitglied muss über alle Teile der Lösungen Auskunft geben können. Verspätete Abgaben werden korrigiert, aber nicht bewertet.

2 Aufgabenstellung

Die Aufgaben beziehen sich auf die Fallstudie aus den Übungen 2 bis 5.

2.1 Funktionsorientiertes Testen (13 Punkte)

Damit Arbeitgeber ein Ticket für ihre Angestellten kaufen können, müssen sie deren AHV-Nummern angeben. Seit 1. Juli 2008 verwenden die zuständigen Behörden in der Schweiz ein neues System für die AHV-Nummern. Sie besteht aus einem dreistelligen Ländercode (756 für die Schweiz), einer neunstelligen Zufallszahl und einer Prüfziffer. Die Prüfziffer wird aus den restlichen Ziffern gemss dem EAN-13 Standard berechnet und dient der Fehlererkennung. Betrachten wir die folgenden Nummer: 123456789012. Die Prüfziffer '2' wird aus den restlichen Stellen '12345678901' errechnet. Ein Tool zur Berechnung der Prüfziffern findet sich unter [2].

Die folgende Methode prüft, ob eine AHV Nummer gemäss [1] gültig ist. Nur Nummern, die genau 16 Ziffern haben und im Format XXX.XXXX.XXXX.XX sind, werden geprüft. Alle anderen Nummern werden als ungültig erkannt.
Ein Beispiel für eine gültige AHV-Nummer ist 756.9217.0769.85.

```
/**
 * This method checks if an AHV number is valid or not.
 *
 * Strings having a length other than 16 or a format other
 * than XXX.XXXX.XXXX.XX are invalid.
 * @param s the AHV number
 * @return true if number is valid, false otherwise
 */

private static boolean checkAHVNumber(String s) throws Exception{
    int length = s.length();
    if (length != 16) {
        return false;
    }
    if (   !( (s.indexOf(".") == 3)
            && (s.indexOf(".", 4) == 8)
            && (s.indexOf(".", 9) == 13)
            && (s.indexOf(".", 14) == -1)
            )
    ){
        return false;
    }
    if( !s.substring(0, 3).equals("756")){
        return false;
    }
    String checkSumString = s.replace(".", "");
    int checkSumLength = checkSumString.length();

    int checkDig = (int) checkSumString.charAt(checkSumLength-1) - (int) '0';
    int sum = 0;
    for (int i = checkSumLength - 2; i >= 0; i -= 2) {
        int digit = (int) checkSumString.charAt(i) - (int) '0';
        if (digit < 0 || digit > 9) {
            throw new Exception("Format Exception");
        }
        sum += digit;
    }
```

```
    sum *= 3;
    for (int i = checkSumLength - 3; i >= 0; i -= 2) {
        int digit = (int) checkSumString.charAt(i) - (int) '0';
        if (digit < 0 || digit > 9) {
            throw new Exception("Format Exception");
        }
        sum += digit;
    }

    return checkDig == 10-(sum % 10);
}
```

a) Entwerfen Sie einen Black-Box-Test für diese Funktion und stellen Sie die von Ihnen ausgewählten Testfälle tabellarisch dar.
   i. Welche Äquivalenzklassen verwenden Sie?
   ii. Welche Grenzfälle entwerfen Sie?
   iii. Schreiben Sie eine Testvorschrift.

b) Testen Sie diese Implementierung mit der Testvorschrift aus Teilaufgabe a) und dokumentieren Sie das Ergebnis (Schreibtischtest ohne Ausführung auf dem Rechner).

c) Berechnen Sie die Zweigüberdeckung Ihres Tests.

d) Wieviele Testfälle benötigen Sie mindestens, um 100% Zweigüberdeckung zu erreichen?

e) Zeigen Sie, dass es Testfälle gibt, welche zusammen 100% Zweigüberdeckung erreichen, ohne dass ein Fehler in `checkAHVNumber` auftritt.

f) Begründen Sie abschliessend generell anhand eines Beispiels, welche Fehlerarten beim White-Box-Test leicht und welche im Gegensatz zum Black-Box-Test schwer erkannt werden können.

[1] `http://www.bsv.admin.ch/themen/ahv/00011/02185/index.html?lang=de`
[2] `http://www.gs1.org/barcodes/support/check_digit_calculator`

2.2 Zielorientiertes Messen (7 Punkte)

Diese Aufgabe bezieht sich auf die Fallstudie zu *iCommute'n'Smile*. Die Software zur Erfassung von Kundendaten und Erstellung von Tickets ist funktionstüchtig und wurde aufgeschalten. Viele Schalterangestellte und Ticketkäufer benutzen sie bereits, obwohl viele von ihnen noch nicht richtig zufrieden mit dem neuen System sind. Die Geschäftsleitung erhält eine beträchtliche Anzahl Beschwerden, dass die Software für viele Anwender zu kompliziert sei. Es scheint, als hätten viele Schaltermitarbeiter zum Erfassen der Daten wieder auf das alte Ticketsystem gewechselt.

Ihr Vorgesetzter beauftragt Sie daher, einen Vorschlag für die Messung des Qualitätsziels "Einfache und effiziente Handhabung von Kundendaten" zu erarbeiten. Verwenden Sie zur Erfüllung dieses Auftrags den GQM-Ansatz aus Kapitel 10, Folie 46 der Vorlesung.

a) Bestimmen Sie mindestens 3 Faktoren (Qualitätsziele) für dieses Ziel und formulieren Sie Fragen, mit denen für jeden Faktor die Erreichung der geforderten Qualität geprüft werden kann. (3 Punkte)

b) Leiten Sie aus den Fragen messbare Merkmale ab. Gehen Sie dabei auch auf die jeweils verwendeten Skalentypen ein. (4 Punkte)