Softwarekonstruktion – Übung 4

4 Qualitätsmanagement und Metriken

4.1 Qualitätsmanagement

4.1.1 Welche zwei grundlegenden Arten von Qualitätsstandards gibt es? Erklären Sie diese kurz.

Produktstandards: definieren Charakteristiken, die alle Produkte aufweisen müssen bei Software z.B. ein gemeinsamer Programmierstil.

Prozessstandards: definieren die Durchführung des Softwareprozesses.

4.1.2 Erläutern Sie den Unterschied zwischen verifizierenden, analysierenden und statisch bzw. dynamisch testenden Prüfverfahren. Nennen Sie jeweils ein Beispiel.

Verifizierende Prüfverfahren haben zum Ziel durch formal exakte Methoden die Konsistenz zwischen (formaler) Spezifikation und Implementierung (im mathematischen Sinne) zu Beweisen. Beispiel: Programmverifikation.

Analysierende Prüfverfahren versuchen über eine Kenngröße Aussagen über die Qualität des Produkts oder Prozesses zu treffen. Beispiel: Softwaremetriken.

Testende Prüfverfahren versuchen mittels einer Stichprobenartigen Überprüfung des zu testenden Programms mit einer gewissen Plausibilität von der korrekten Funktionsweise des Programms zu überzeugen. Es verbleibt aber immer ein Rest Unsicherheit. Dabei wird zwischen dynamischen Testverfahren, die den Code zum Testen ausführen, z.B. kontrollflussbezogene Überdeckungstests und statischen Testverfahren, wie bspw. Audits und Reviews, die den Code nicht ausführen.

4.2 Standards im Qualitätsmanagement

4.2.1 Was versteht man unter ISO9001 und CMMI? Welche Ziele verfolgen sie?

ISO 9001 ist ein Prozessqualitätsstandard und beschreibt ganz allgemein Modelle zur Darlegung der Qualitätssicherung in Entwicklung, Produktion, Montage und Kundendienst. Das Capability Maturity Model Integrated gibt Hilfestellung bei der Bewertung und Verbesserung des Qualitätsmanagements.

4.2.2 Welchen der beiden Standards würden Sie nutzen für?

1. Dokumentation, Überprüfbarkeit oder Transparenz von bestehenden Qualitätsprozessen
2. Bewertung oder Verbesserung des Qualitätsmanagements

1. ISO 9001
2. CMMI

4.3 Zyklomatische Komplexität

4.3.1 Geben Sie an, wie sich Sequenzen, Verzweigungen und Sprünge auf die zyklomatische Komplexität auswirken.

Sequenzen können beliebig lang sein ohne die zyklomatische Komplexität zu erhöhen.(Kann man beim zeichnen von Kontrollflussgraphen ausnutzen und Sequenzen zu einem Knoten zusammenfassen.
Jede Verzweigung in einer Auswahl erhöht die zyklomatische Komplexität um 1.
Goto Anweisungen erhöhen die zyklomatische Komplexität nicht.

4.3.2 Zeichnen Sie den Kontrollflussgraphen zu folgender Funktion und bestimmen Sie ihre zyklomatische Komplexität.

```
void output( int selection, int y){
        if (selection ==1){
                printf("\n 1");
        }
        else {
                if (selection ==2) {
                        printf("\n 2");
                }
                else{
                        if (selection ==3){
                                printf("\n 3");
                        }
                        else{
                                if (selection ==4){
                                        if (y == 0){
                                                printf("\n 4");
                                                printf("\n 5");
                                        }
                                        else{
                                                printf("\n 6");
                                        }
                                }
                                else{
                                        printf("\n 7");
                                }
                        }
                }
        }
}
```

Die Funktion hat eine zyklomatische Komplexität von 6