Universität des Saarlandes
Lehrstuhl für Softwaretechnik
Prof. Andreas Zeller, Holger Cleve

2. Übung zu Softwaretechnik 2

Generative Programmierung

1 Grundlagen Management

1.1 Management-Strategien

Benennen Sie die drei wesentlichen Managementstrategien. Welche Auswirkungen haben diese auf den Software-Entwicklungsprozeß?

1.2 Produktivität

Welche der folgenden Änderungen in der Projektsituation steigert die Produktivität am meisten

- Das Produkt, das Sie entwickeln, ist nicht mehr komplex, sondern es ist als einfach einzustufen.
- Sie haben CASE-Werkzeuge über einen längeren Zeitraum eingeführt.
- Ihr Team besteht, anstatt aus durchschnittlichen Mitarbeitern, aus den besten Mitarbeitern der Firma.
- Das Großraumbüro wurde abgeschafft, und alle ihre Mitarbeiter haben Einzelzimmer.

1.3 Projektleitung

Sie werden Projektleiter eines kleinen Entwicklungsteams. Ihr Team entwickelt Software in der Programmiersprache C. Bisher bekam das Team einen mündlichen Auftrag, eine Software für einen Kunden zu entwickeln, setzte sich dann zusammen, teilte die Aufgabenbereiche auf und fing an zu entwickeln. Schnittstellenabsprachen wurden direkt zwischen den Programmierern erledigt, da diese alle zusammen in einem Raum arbeiteten. War das Produkt fertig, wurde es ausgeliefert, und das Team realisierte anschließend eventuelle Änderungen aufgrund von Kundenwünschen. Sie sollen langfristig die Produktivität steigern. Welche Maßnahmen führen Sie durch?

2 Generative Programmierung

Die folgenden Aufgaben sollen so abstrakt wie möglich realisiert werden: Hinreichend um das Prinzip zu verdeutlichen (und mit gcc auszuprobieren) und so wenig wie Implementierungsaufwand (also keine Fensteroberfläche etc.). Schreiben Sie die Templates in C++ und probieren Sie sie mit dem GCC aus.

2.1 Templates

Entwerfen Sie (analog zu Vector und Stack in der Vorlesung) ein Template für eine Queue. Die Queue soll die Funktionen isempty, add, first und remove haben.

2.2 Spezialisierung

Eine Prioritätswarteschlange fügt ein Element nicht einfach am Ende ein, sondern vor allen anderen Elementen mit der gleichen Priorität. Erweitern Sie Ihr Queue-Template so, daß es für alle von einer abstrakten Klasse Comparable abgeleitete Funktionen eine Prioritätswarteschlange ist, für alle anderen eine normale First-Come – First-Serve Queue.

2.3 Metaprogramming

```
int fastpot(int n, int x) {
  if(n==0)
    return 1;
  else if (n%2 == 0)
    return sqr(fastpot(n/2,x));
  else
    return x * fastpot(n-1,x);
}
```

Abbildung 1: Algorithmus zur schnellen Potenzierung

Implementieren Sie den Algorithmus zur schnellen Potenzberechnung aus Abbildung 1, der x^n berechnet, als ein Template, das mit der Potenz instanziiert wird. Das Template berechnet zur Compilezeit, wie oft die Eingabe quadriert werden soll.

- n wird als Integer (ab 1), x als allgemeiner Typ übergeben.
- Das Template ist ein Funktionsobjekt mit der Funktion `T operator() (T x) {...};`
- Hilfstemplates bestimmen, welche Berechnung ausgeführt wird.

Beispiel: `FASTPOT<5> pot5; pot5(2);` berechnet $2^5 = 32$.

Fragen?

Bei Fragen wenden sie sich an Holger Cleve `<cleve@cs.uni-sb.de>`