Philipps-Universität Marburg
Fachbereich Mathematik
Fachgebiet Informatik

Informatik-Fortgeschrittenen-Praktikum

Richtlinien für die Erstellung der Dokumentation

Für jede bearbeitete Praktikumsaufgabe ist eine Dokumentation zu erstellen und zum Abschluß der Arbeit vorzulegen. Die Annahme der Dokumentation durch den/die Betreuer ist Voraussetzung für die Ausgabe des Praktikumsscheins. Es empfiehlt sich, die Dokumentation nicht erst nach Lösung der Aufgabe und Fertigstellung des Programms zu beginnen, sondern sie schon bei Beginn der Arbeit anzulegen und während der Arbeit laufend fortzuschreiben.

A Allgemeiner Aufbau

Die Dokumentation sollte folgendermaßen aufgebaut sein:

- Deckblatt

  (enthält Titel der Aufgabe, Semesterangabe, Namen der Bearbeiter, Namen der (des) Betreuer(s), Beginn- und Abgabedatum)

- Inhaltsverzeichnis

  (bezieht sich auf den folgenden Hauptteil)

- Hauptteil (vgl. unten)

- Literaturverzeichnis

- Anhänge (soweit erforderlich)

B Aufbau des Hauptteils

Im Hauptteil sind die Problemstellung und das erarbeitete Ergebnis systematisch und im Zusammenhang darzustellen. Zwischenschritte und versuchsweise beschrittene Lösungswege sind nur dann zu dokumentieren, wenn sie für das Verständnis oder zur Begründung des eingeschlagenen Lösungsweges wichtig sind oder anderweitig von bleibendem Interesse sind.

In seinem Aufbau sollte sich der Hauptteil an der Dokumentationsstruktur für normale Software-Projekte orientieren, d.h. jede Aufgabe wird wie ein kleines Projekt behandelt. In Analogie zu größeren Projekten nennen wir das zu erstellende Produkt "Software-System" oder kurz "System".

Software-Projekte beginnen in der Regel aufgrund eines Projekt-Vorschlags oder -Auftrags, der die Aufgabenstellung und Randbedingungen umreißt und durchlaufen dann die folgenden Abschnitte:

- Problemanalyse und Festlegung der Anforderungen

- Fachkonzept (Fachlicher Entwurf)

- DV-technischer Entwurf

- Realisierung (Programmierung, Test und Integration)

- Installation und Einsatz

Dementsprechend empfiehlt sich folgender Aufbau des Hauptteils (der selbstverständlich aufgrund spezifischer Erfordernisse der Aufgabenstellung ggf. zu modifizieren oder zu ergänzen ist).

1 Aufgabenstellung und Problemanalyse

Zunächst ist die gestellte Aufgabe kurz zu umreißen, dazu die vorher festgelegten Rahmenbedingungen wie z.B. Basissystem, Programmiersprache, vorgegebene Werkzeuge etc..

Soweit die Anforderungen an das zu erstellende System noch nicht endgültig festliegen, sind diese im Gespräch mit dem Betreuer und ggf. potentiellen Benutzern zu klären und das Ergebnis ist schriftlich festzuhalten.

Problemanalyse bedeutet, die gestellte Aufgabe und die gestellten Anforderungen gedanklich zu durchdringen. Dazu ist es wichtig, die Gegenstände, auf die sich die Aufgabe bezieht und die Funktionen, die programmtechnisch zu realisieren sind, zunächst aus der Sicht eines potentiellen Anwenders zu identifizieren.

Falls mehrere Lösungsalternativen bestehen, sind diese zu diskutieren und - soweit relevant - zu dokumentieren. Die gewählte Lösungsidee ist darzustellen und (falls Alternativen bestanden), die Wahl zu begründen.

Die Darstellungsform ist vorzugsweise Text, aber auch graphische Darstellungen wie z.B. Datenflußdiagramme (DFD’s), Zustandsübergangsdiagramme (endliche Automaten) können hilfreich sein.

2 Fachkonzept

Hier sind die relevanten Gegenstände und Funktionen des Systems aus fachlicher Sicht im Detail und mit ihren wechselseitigen Beziehungen zu beschreiben. Bei größeren Software-Projekten ist es heute üblich, dafür ein Daten-, ein Funktions- und ein Ablaufmodell zu erstellen. Im Datenmodell werden die relevanten Gegenstände, im Funktionsmodell die zu realisierenden Funktionen (bezogen auf die Gegenstände) beschrieben. Beide können auch im Rahmen eines objektorientierten Anwendungsmodells zusammengefaßt werden. Im Ablaufmodell werden die gegenstands-übergreifenden dynamischen Zusammenhänge dargestellt. Dazu gehört ggf. auch die Beschreibung der Benutzer-Schnittstelle, z.B. in Form eines Prototyps.

Für die Praktikums-Dokumentation empfiehlt sich eine analoge Struktur, wobei je nach Aufgabenstellung des Gewicht des Daten- oder des Funktionsmodells überwiegen kann. Im Falle eines Textverarbeitungsprogramms wären z.B. relevante Gegenstände Texte, Abschnitte, Überschriften, Sätze, Worte oder Zeichen. Funktionen wären Einfügen, Löschen, Umstellen, Suchen, Drucken etc..

Bekannte Darstellungsformen sind neben Text Entity-/Relationship-Diagramme (für das Datenmodell), Datenflußdiagramme, SADT-Diagramme oder HIPO-artige Tabellen (für funktionale Zusammenhänge), Klassen-/Objektdiagramme (etwa nach Coad/Yourdon oder in UML – Unified Modeling Language) sowie Sequenzdiagramme für Abläufe.

3 Technischer Entwurf

Dieser Abschnitt stellt die Struktur des zu erstellenden Systems aus Software-technischer Sicht dar. Er gliedert sich in:

- Gesamtübersicht (statische und dynamische Systemstruktur),

- Übergreifende Konzepte und

- Spezifikation der Systembausteine (Module/Klassen und ggf. Komponenten/Subsysteme)

Die *Gesamtübersicht* zeigt die Bausteine der Systemzerlegung (Module oder units bzw. Klassen) im Zusammenhang und wird am besten durch ein Diagramm veranschaulicht. Gibt es eine größere Anzahl von Modulen, dann kann eine Zusammenfassung logisch zusammengehöriger Teile zu Komponenten sinnvoll sein.

Neben der statischen Systemstruktur (Zerlegung des Systems in Komponenten und Module) ist vor allem die dynamische Struktur wichtig, die die möglichen "Benutzt"-Beziehungen des Module untereinander aufzeigt. Daraus ergibt sich häufig eine Schichtenstruktur: Module höherer Schichten benutzen solche darunterliegender Schichten, nicht aber umgekehrt. Benutzt-Beziehungen innerhalb der gleichen Schicht oder über zwei Schichtengrenzen hinweg sind zugelassen.

In den Abschnitt *Übergreifende Konzepte* gehören Entwurfsentscheidungen, die das ganze System betreffen, wie z.B. Wahl und Abgrenzung der Schichten, Fehlerbehandlung oder global benutzte Datenstrukturen oder Systemfunktionen. Daneben sind hier durchgängig benutzte Standards wie z.B. Namenskonventionen, der Gebrauch von Return- oder Fehlercodes etc. zu beschreiben.

Den Kern des technischen Entwurfs bilden die *Spezifikationen* der einzelnen Module. Diese sind folgendermaßen aufgebaut:

- Modulname, (statische und dynamische) Einordnung in das Gesamtsystem

- Kurzbeschreibung der durch den Modul gelösten Teilaufgabe

- Export-Schnittstelle des Moduls

Diese enthält alle Operationen (i.d.R. realisiert durch Prozeduren oder Funktionen) und möglicherweise Datenstrukturen, die von dem beschriebenen Modul exportiert, d.h. anderen Modulen zur Benutzung zur Verfügung gestellt werden. Prozeduren/Funktionen werden durch Angabe ihres Namens, Anzahl, Namen und Typen der Parameter und Ergebnisse und eine kurze Beschreibung ihrer Bedeutung ("Semantik") spezifiziert. Dabei ist das Auftreten möglicher Fehlersituationen und die Reaktion auf Fehler zu berücksichtigen.

- Import-Schnittstelle des Moduls

Diese enthält die Liste aller vom aktuellen Modul zu benutzenden Module ("uses-part") und für jeden dieser Module die Aufzählung der benutzten Operationen und/oder Datenstrukturen.

4 Realisierung (Programmierung, Test und Integration)

Für die Fein-Dokumentation der Programme genügen in der Regel systematisch und diszipliniert angebrachte Kommentare. So kann z.B. ein Kommentarkasten am Beginn jeder Prozedur bzw. Funktion zweckmäßig sein, der kurz Sinn, Zweck und Funktionsweise beschreibt. Ansonsten gilt für Kommentare:

- Kommentare möglichst gut als solche erkennbar anbringen (Standard-Layout!) und dort, wo sie wirklich hingehören.

- Nur das Notwendige und das kurz und bündig kommentieren. Ein Kommentar wie der folgende zu

  A := A+1 -- Hier wird A um eins erhöht

  erübrigt sich !

- Leitgedanke für jeden Kommentar sollte sein: Was erwartet ein fremder Entwickler, der das Programm noch nicht kennt und möglichst schnell verstehen möchte?

Zu einem fertig entwickelten Programm gehören auch systematisch durchgeführte *Tests*. Dafür werden *Testfälle* entworfen, die die Funktionalität des Programms möglichst vollständig abdecken und mit repräsentativ ausgewählten *Testdaten* zum Ablauf gebracht werden. Achtung: Test und Fehlersuche ("debugging") sind zwei verschiedene Paar Stiefel! Fehlersuche braucht (und soll) nicht dokumentiert werden, die Anlage und Durchführung von Tests dagegen auf jeden Fall. Leitlinie für die Testdokumentation sollte sein, daß ein Fremder den Test aufgrund der Dokumentation jederzeit wiederholen könnte.

Integration heißt, die entwickelten Bausteine zum Gesamtsystem zusammenzufügen. Werden dabei keine Zwischenstufen ("Subsysteme") gebildet, so genügt für die Dokumentation ein Rückgriff auf die Gesamtübersicht am Beginn von Abschnitt 3. Für durchgeführte Integrationstests gilt das im vorigen Abschnitt Gesagte sinngemäß.

5 Benutzer-Dokumentation

Diese enthält alle Angaben, die für Uneingeweihte (d.h. an der Systementwicklung nicht Beteiligte) notwendig sind, um

a) das System zu installieren,

b) das System zu benutzen.

Zur Dokumentation gehört damit

- eine Installations-Diskette mit Anweisungen für den Installationsvorgang und ggf. dabei zu beachtende Randbedingungen,

- ein Benutzer-Handbuch, das die Benutzer-Schnittstelle im Zusammenhang darstellt, die einzelnen Benutzerfunktionen mit Eingaben (z.B. Masken, Menüs), Ausgaben (z.B. Bildschirmausgaben, Listen) sowie deren Handhabung und ggf. zu beachtende Nebenbedingungen beschreibt.