Anforderungsdefinition

Carl von Ossietzky Universität Oldenburg
Department Informatik
Abteilung Software Engineering

Individuelles Projekt - Anhang
Fallstudie "File Manager"

Name: Stefan Gudenkauf
Matr.Nr.: 7770870

Beurteilender Hochschullehrer: Prof. Dr. Wilhelm Hasselbring
Zweitgutachter: Jun.-Prof. Dr. Ralf H. Reussner

Ort, Datum: Oldenburg, im November 2004

Inhaltsverzeichnis

Dieses Dokument stellt eine exemplarische Anforderungsdefinition für die Entwicklung eines *File Managers* dar und ist im Rahmen des Individuellen Projektes *"Entwicklung eines Anwendungsbeispiels für die Ausbildung im Software Engineering mittels Eclipse-UML"* von ***Stefan Gudenkauf*** entstanden.

Die Anforderungsdefinition kann und soll zusammen mit der Fallstudie in der Lehre Verwendung finden. Sie soll als Anschauungsbeispiel und Vorlage für Anforderungsdefinitionen von Software dienen.

1 Ausgangssituation und Zielsetzung

Im Zuge des Individuellen Projektes *"Entwicklung eines Anwendungsbeispiels für die Ausbildung im Software Engineering mittels EclipseUML"* soll ein Dateiverwaltungsprogramm, ein sogenannter *File Manager*, entwickelt werden. Dieser wird als typischer *File Manager* den Zugriff auf ein Dateisystem verwalten sowie das Öffnen, Schließen, Lesen und Schreiben von Dateien ermöglichen. Des weiteren wird er bekannte und esssentielle *Software Engineering*-Konzepte beinhalten und so ein Lehrbeispiel für die *Software Engineering*-Ausbildung darstellen.

Zur Realisierung erfolgt die Implementation in *Java* und als Entwicklungsplattform wird *Eclipse* verwendet. Insbesondere kommt bei der Entwicklung das *Eclipse*-PlugIn *EclipseUML* zum Einsatz. Ebenfalls wird der *File Manager* modular aufgebaut, um zukünftigen Erweiterungen und Modifikationen gegenüber offen zu sein.

Weiterhin werden folgende Aspekte realisiert, die der *File Manager* gewährleisten soll:

- Der *File Manager* wird eine bedienfreundliche Benutzeroberfläche besitzen, die dem Benutzer eine möglichst angenehme Umgebung für die Arbeit mit dem *File Manager* bereitstellt.
- Der *File Manager* wird eine effiziente und praktische Benutzeroberfläche besitzen, die dem Benutzer alle Funktionalitäten des *File Managers* klar und übersichtlich zur Verfügung stellt.
- Die einzelnen Systemebenen werden voneinander getrennt: Es wird zwischen grafischer Benutzeroberfläche und Anwendungsebene unterschieden.
- Die Systemkonsistenz soll gewährlestet werden. Insbesondere bei Nutzertätigkeit und gegebenfalls Fehlerausgabe bei Verletzungen der Konsistenzbedingungen.

Als Vorgehensmodell für die Entwicklung kann ein iteratives evolutionäres Wasserfallmodell verwendet werden, das Rückschritte im Entwicklungsprozess zulässt.

2 Systemeinsatz und Systemumgebung

Der *File Manager* wird zunächst auf modular entwickelt. Um später eine Portierung auf andere Systeme zu erleichtern erfolgt die Implementierung in *Java*, da diese Programmiersprache eine breite Palette an Bibliotheken bereitstellt und plattformunabhängig ist. Die Ansprache eines möglichst großen Kunden- bzw. Nutzerkreises ist damit ebenfalls gesichert. Die Zielsysteme sind die Rechner des Lehrenden und der Teilnehmer des Studienmoduls 'Software Engineering' an der Carl von Ossietzky Universität Oldenburg.

Auf den Zielsystemen wird vorausgesetzt:

- *Java*-Entwicklungsumgebung (*JDK* oder *J2SE*) in aktueller Version mit *Java Swing* Unterstützung.

Zudem wird vorausgesetzt, dass die Rechenleistung zur Realisierung der implementierten Algorithmen ausreichend ist und genügend Speicherkapazizät vorhanden ist, um die Anwendung ausführen zu können.

Die Benutzung des Systems soll durch die grafische Oberfläche weitgehend intuitiv sein. Daher müssen die EDV-Kenntnisse der Benutzer nicht allzu hoch ausfallen. Ausreichende EDV-Kenntnisse sollten ohnehin vorhanden sein, da die Zielgruppe stellen die Teilnehmer des Studienmoduls "Software Engineering" an der Carl von Ossietzky Universität dar.

3 Funktionale Anforderungen

3.1 Beschreibung der Daten und Datenhaltung

Ein *File Manager* benötigt prinzipiell keine Verbindung zu einer Datenbank. Er verwaltet lediglich Datenstrukturen, anstatt Daten halten zu müssen. Daher entfällt eine Beschreibung der Datenhaltung.

3.2 Programmstrukur

Der *File Manager* wird nicht zwischen verschiedenen Benutzertypen wie Administratoren und "normalen Benutzern" unterscheiden. Zudem besteht das *File Manager*-System nur aus einem einzelnen Programm. Aus diesem Grund besteht die einzige Schnittstelle zum Benutzer aus einer einzelnen grafischen Bedienoberfläche, die keine verschiedenen Sichten für verschiedene Benutzergruppen bereitstellen muss.

Der Benutzer gebraucht das System ausschließlich zur Verwaltung eines Dateisystems und zum Öffnen, Schließen, Lesen und Schreiben von Dateien. Weitere Funktionen sind bisher nicht vorgesehen, können jedoch zu einem späteren Zeitpunkt leicht hinzugefügt werden, da der *File Manager* modular entwickelt wird.

Um der Forderung nach Modularität, Erweiterbarkeit und Modifizierbarkeit gerecht zu werden, wird der Quellcode in verschiedene Pakete unterteilt geben. Das Hauptpaket `entitymanager` wird dann folgende Unterpakete beinhalten:

- `logic`: Klassen für die Anwendungsebene; Unterpackage `file` zur Dateibehandlung;
- `presentation`: Klassen für die grafische Darstellung; Unterpackage `swing` für *Java Swing*-Komponenten;

Damit wird gewährleistet, dass die Anwendungslogik oder die grafische Benutzeroberfläche ausgetauscht werden kann. So kann zum Beispiel durch den Austausch oder die Modiifikation der Anwendungslogik statt der Repräsentation der Dateisysteme eine Repräsentation von *CVS-Repositories* erfolgen oder die verwendete *Java Swing*-GUI durch eine reine *Java AWT*-GUI ersetzt werden.

Folgendes Paketdiagramm stellt die beschriebene Einteilung grafisch dar:

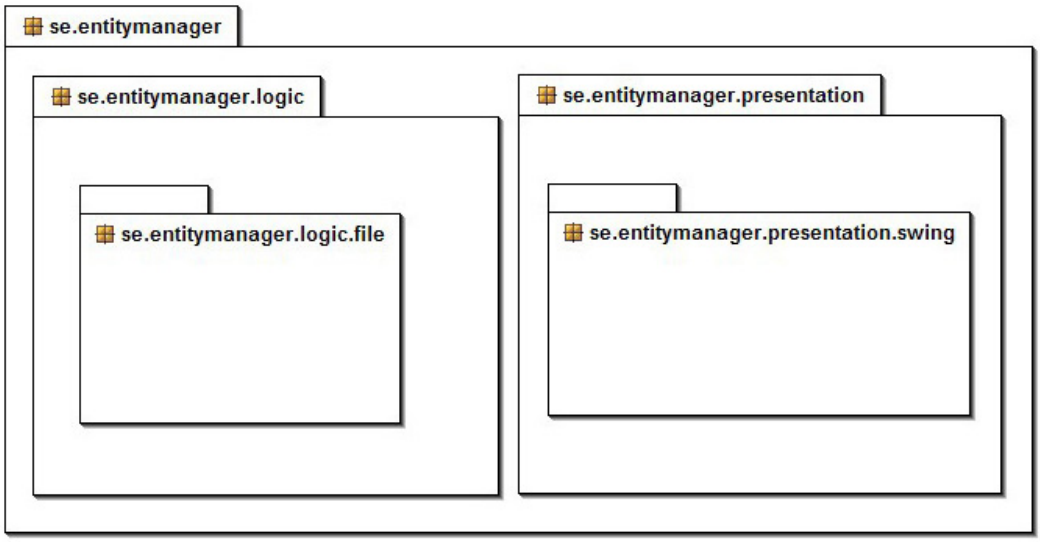

Abb. 3.1 Paketdiagramm des File Manager

3.3 Funktionalität

Das System wird primär in der Lage sein, Dateisysteme zu verwalten. Das beinhaltet unter anderem die Darstellung des Dateisystems, das Navigieren und Durchsuchen des Dateisystems sowie das Öffnen, Schließen, Lesen und Schreiben, Verschieben und Umbenennen von Dateien.

Diese Funktionen werden durch das folgendes Anwendungsfalldiagramm beschieben. Anschließend werden die einzelnen Anwendungsfälle detailliert wiedergegeben:

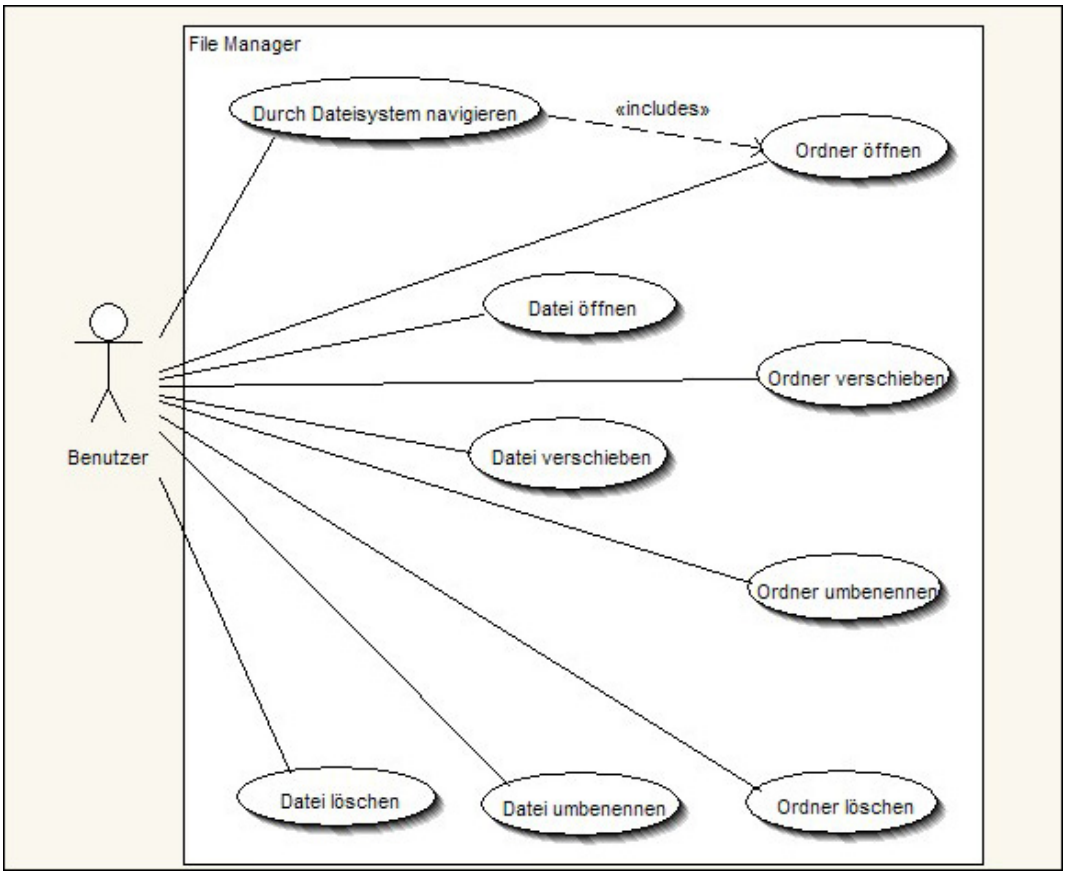

Abb. 3.2 Anwendungsfalldiagramm des File Manager

Beschreibung der einzelnen Anwendungsfälle:

ANWENDUNGSFALL
Ordner öffnen

ZWECK
Öffnen eines Ordners im Dateisystem
AKTEURE
Benutzer
EREIGNIS
Benutzer will einen Ordner im Dateisystem öffnen und dessen Inhalt anzeigen lassen
VORBEDINGUNG
Ordner vorhanden
RELEVANTE EINGABEDATEN
Änderungen
AUSNAHMEN
mögliche Fehlerfälle:
Ordner nicht vorhanden

ERGEBNIS
Ordner wurde geöffnet und dessen Inhalt wird dargestellt
NACHBEDINGUNG
Ansicht muss aktualisieren und Inhalt des Ordners anzeigen
SZENARIO
1. Ordner in der Verzeichnisdarstellung markieren (optional)
2. Ordner in der Verzeichnisdarstellung öffnen
3. Darstellung aktualisieren und Ordnerinhalt darstellen

ANWENDUNGSFALL
Ordner verschieben

ZWECK
Verschieben eines Ordners im Dateisystem
AKTEURE
Benutzer
EREIGNIS
Benutzer will einen Ordner im Dateisystem verschieben
VORBEDINGUNG
Ordner vorhanden
RELEVANTE EINGABEDATEN
Änderungen
AUSNAHMEN
mögliche Fehlerfälle:
Ordner nicht vorhanden; Zielort der Verschiebung ungültig

ERGEBNIS
Ordner wurde an einen anderen Ort im Dateisystem verschoben
NACHBEDINGUNG
Ordner ist an Ursprungsort nicht mehr vorhanden; Ordner ist an Zielort vorhanden; Inhalt des Ordners wurde vollständig kopiert
SZENARIO
1. Ordner in der Verzeichnisdarstellung markieren (optional)
2. Ordner ausschneiden
3. In der Verzeichnisdarstellung an Zielort navigieren
4. Zielort markieren
5. Ordner an Zielort einfügen
6. Ordner und Ordnerinhalt werden an Zielort kopiert und an Ursprungsort gelöscht
7. Änderung in Dateisystem und Verzeichnisdarstellung übernehmen

ANWENDUNGSFALL
Ordner umbenennen

ZWECK
Umbenennen eines Ordners im Dateisystem
AKTEURE
Benutzer
EREIGNIS
Benutzer will einen Ordner im Dateisystem umbenennen
VORBEDINGUNG
Ordner vorhanden
RELEVANTE EINGABEDATEN
Änderungen
AUSNAHMEN
mögliche Fehlerfälle:
Ordner nicht vorhanden; neuer Name des Ordners bereits im selben Verzeichnis vorhanden

ERGEBNIS
Ordner wurde umbenannt
NACHBEDINGUNG
Ordner ist im Verzeichnis nicht mehr unter altem Namen erreichbar; Ordner ist im Verzeichnis unter neuem Namen erreichbar; Ordner enthält weiterhin seinen ursprünglichen Inhalt
SZENARIO
1. Ordner in der Verzeichnisdarstellung markieren (optional)
2. Ordner umbenennen
3. Änderung in Dateisystem und Verzeichnisdarstellung übernehmen

ANWENDUNGSFALL
Ordner löschen

ZWECK
Löschen eines Ordners im Dateisystem
AKTEURE
Benutzer
EREIGNIS
Benutzer will einen Ordner im Dateisystem löschen
VORBEDINGUNG
Ordner vorhanden
RELEVANTE EINGABEDATEN
Änderungen
AUSNAHMEN
mögliche Fehlerfälle:
Ordner nicht vorhanden; Ordner schreibgeschüzt

ERGEBNIS
Ordner wurde gelöscht
NACHBEDINGUNG
Ordner und sein gesamter Inhalt sind nicht mehr im Dateisystem vorhanden
SZENARIO
1. Ordner in der Verzeichnisdarstellung markieren (optional)
2. Ordner (inklusive Inhalt) löschen
3. Änderung in Dateisystem und Verzeichnisdarstellung übernehmen

ANWENDUNGSFALL
Datei öffnen

ZWECK
Öffnen einer Datei im Dateisystem
AKTEURE
Benutzer
EREIGNIS
Benutzer will eine Datei im Dateisystem öffnen
VORBEDINGUNG
Datei vorhanden; Datei kann vom File Manager dargestellt werden oder File Manager kann Öffnungsaufruf an ein externes Programm vermitteln
RELEVANTE EINGABEDATEN
Datei
AUSNAHMEN
mögliche Fehlerfälle:
Datei nicht vorhanden; Datei kann nicht dargestellt werden

ERGEBNIS
Datei wird entweder durch den File Manager geöffnet oder (durch ein externes) Programm dargestellt
NACHBEDINGUNG
keine
SZENARIO
1. Datei in der Verzeichnisdarstellung markieren (optional)
2. Datei in der Verzeichnisdarstellung öffnen
3a. Datei wird durch den File Manager dargestellt (z.B. ASCII-Textdateien)
3b. File Manager ruft externes Programm zum Öffnen der Datei auf

ANWENDUNGSFALL
Datei verschieben

ZWECK
Verschieben einer Datei im Dateisystem
AKTEURE
Benutzer
EREIGNIS
Benutzer will eine Datei im Dateisystem verschieben
VORBEDINGUNG
Datei vorhanden
RELEVANTE EINGABEDATEN
Änderungen
AUSNAHMEN
mögliche Fehlerfälle:
Datei nicht vorhanden; Zielort der Verschiebung ungültig

ERGEBNIS
Datei wurde an einen anderen Ort im Dateisystem verschoben
NACHBEDINGUNG
Datei ist am Ursprungsort nicht mehr vorhanden; Datei ist am Zielort vorhanden
SZENARIO
1. Datei in der Verzeichnisdarstellung markieren (optional)
2. Datei ausschneiden
3. In der Verzeichnisdarstellung an Zielort navigieren
4. Zielort markieren
5. Datei an Zielort einfügen
6. Datei wird an den Zielort kopiert und am Ursprungsort gelöscht
7. Änderung in Dateisystem und Verzeichnisdarstellung übernehmen

ANWENDUNGSFALL
Datei umbenennen

ZWECK
Umbenennen einer Datei im Dateisystem
AKTEURE
Benutzer
EREIGNIS
Benutzer will eine Datei im Dateisystem umbenennen
VORBEDINGUNG
Datei vorhanden
RELEVANTE EINGABEDATEN
Änderungen
AUSNAHMEN
mögliche Fehlerfälle:
Datei nicht vorhanden; neuer Name der Datei bereits im selben Verzeichnis vorhanden

ERGEBNIS
Datei wurde umbenannt
NACHBEDINGUNG
Datei ist im Verzeichnis nicht mehr unter altem Namen erreichbar; Datei ist im Verzeichnis unter neuen Namen erreichbar
SZENARIO
1. Datei in der Verzeichnisdarstellung markieren (optional)
2. Datei umbenennen
3. Änderung in Dateisystem und Verzeichnisdarstellung übernehmen

ANWENDUNGSFALL
Datei löschen

ZWECK
Löschen einer Datei im Dateisystem
AKTEURE
Benutzer
EREIGNIS
Benutzer will eine Datei im Dateisystem löschen
VORBEDINGUNG
Datei vorhanden
RELEVANTE EINGABEDATEN
Änderungen
AUSNAHMEN
mögliche Fehlerfälle:
Datei nicht vorhanden; Datei schreibgeschüzt

ERGEBNIS
Datei wurde gelöscht
NACHBEDINGUNG
Datei und sein gesamter Inhalt sind nicht mehr im Dateisystem vorhanden
SZENARIO
1. Datei in der Verzeichnisdarstellung markieren (optional)
2. Datei löschen
3. Änderung in Dateisystem und Verzeichnisdarstellung übernehmen

4 Nichtfunktionale Anforderungen

Da als Endprodukt ein konkurrenzfähiges Softwareprodukt entwickelt werden soll, sind an das Gesamtsystem Anforderungen gestellt, die unter allen gegebenen Umständen berücksichtigt werden müssen.

Dazu gehören unter anderem:

- Benutzerfreundlichkeit;
- Geringer Wartungs- und Installationsaufwand;
- Möglichkeit zum Neustart nach Ausfall;
- Ausfallsicherheit;
- Systemeffizienz;

Die Arbeit mit dem *File Manager* soll möglichst angenehm und störungsfrei erfolgen. Dazu müssen alle notwendigen Berechnungen und Operationen in angemessener Zeit erfolgen, damit die Systemeffizienz nicht beeinträchtigt und die Geduld des Nutzers nicht unnötig belastet wird.

Aus denselben Gründen muss das System möglichst ausfallsicher sein. Sollte der *File Manager* dennoch ausfallen, so müssen getroffene Einstellungen und Bearbeitungen gegebenenfalls wiederhergestellt und die Arbeit problemlos fortgesetzt werden können. Dies beinhaltet ebenfalls den problemlosen Neustart des Systems.

Außerdem muss der *File Manager* einfach zu warten und zu installieren sein. Die Wartung umschließt u.a. die Möglichkeit, neue Funktionen mit geringem Aufwand integrieren zu können. Dieses wird durch den modularen Aufbau des Systems unterstützt. Eine einfache Installation ist schon deshalb ein notwendiger Bestandteil der Entwicklung, da sie auch für unterschiedliche Systeme eine schnelle Einrichtung des *File Managers* sicherstellen soll.

Zuletzt sei noch die Wichtigkeit der Benutzerfreundlichkeit betont, da ohne eine benutzerfreundliche grafische Oberfläche ebenfalls kein effizientes Arbeiten möglich ist. Zudem kann durch eine undurchdachte Benutzeroberfläche die Frustration schnell so hoch ansteigen, dass das gesamte System - unabhängig von den bereitgestellten Funktionalitäten - von den Benutzern abgelehnt wird.

5 Benutzerschnittstelle

Der *File Manager* wird die Verzeichnis- und Dateistrukturen des Anwendersystems darstellen und verwalten können. Dazu soll die Bedienung über eine fensterorientierte Oberfläche gehandhabt werden. Hauptbestandteil des Programmfensters wird eine doppelte Ansicht der Verzeichnis- und Dateistrukturen in Baumansicht sein. Übersichtlichkeit, klare Strukturen und Ergonomie sollen zur leichten Auffassung der Fülle an Informationen beitragen.

Folgende Funktionen sollen dabei unterstützt werden:

- Ordner öffnen
- Ordner umbenennen
- Ordner verschieben
- Ordner löschen
- Datei öffnen
- Datei umbenennen
- Datei verschieben
- Datei löschen

Eine mögliche grafische Darstellung könnte wie folgt aussehen:

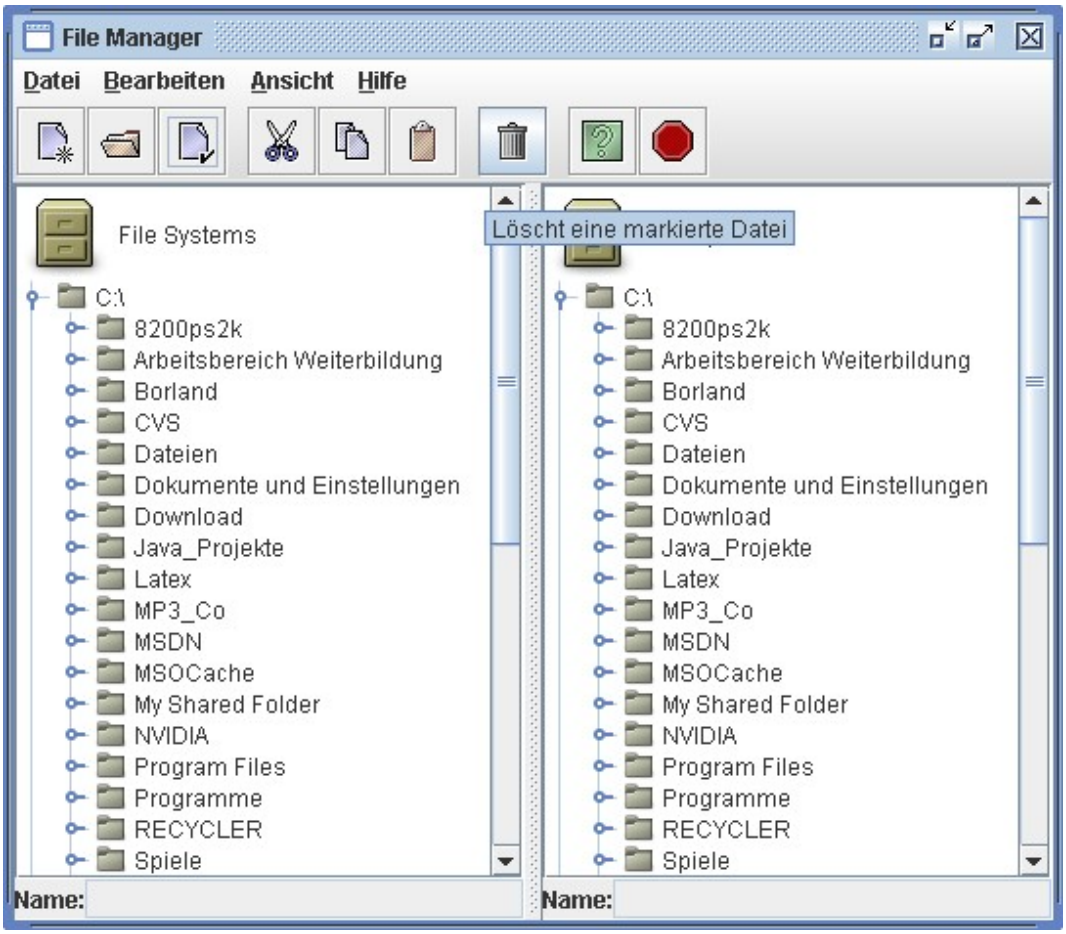

Abb. 5.1 Exemplarische grafische Benutzeroberfläche des File Manager

Um den Anforderungen nach Benutzerfreundlichkeit und Funktionalität gerecht werden zu können, empfiehlt es sich, eine "klassische" Programmoberfläche zu verwenden: Es wird eine Menüleiste über die Breite des Programmhauptfensters verlaufen, die alle Funktionalitäten anbietet. Dabei kann die typische Menüorganisation mit Menüoberpunkten wie "Datei", "Bearbeiten", "Einstellungen", "Hilfe" etc. realisiert werden. Nicht intuitive Menüketten werden vermieden.

Darunter wird sich die doppelte Verzeichnisansicht befinden. Sie wird den größten Platz im Programmfenster einnehmen und die Verzeichnis- und Dateistruktur des Anwendersystems in Baumstruktur visuell wiedergeben.

Unter der Verzeichnisansicht bietet es sich an, die am häufigsten benutzten Kernfunktionen in Form von Schaltflächen bereitzustellen. Mögliche Schaltflächen sind beispielsweise die Funktionen "Datei öffnen" und "Datei ausschneiden".

6 Fehlertests und Fehlerverhalten

Sobald ein erster lauffähiger Prototyp des *File Managers* vorliegt, werden in regelmäßigen Abständen Tests erfolgen. Das Programm wird hierzu auf verschiedenen Zielsystemen, z.B. *FreeBSD*, *Sun*, *Windows* etc. installiert und anschließend durch sinnvolle und unsinnige Eingaben abgetestet. Dabei werden so viele Anwendungsfälle wie möglich abgedeckt.

Daneben werden auch systemspezifische Probleme bedacht, wie etwa eine eventuelle begrenzte Farbdarstellung oder die ungewöhnlichen Schriftgrössen der *Sun*-Systeme. Ein Einsatz von *JUnit* oder ähnlichem ist nur in geringem Maße vorgesehen, da der Aufwand für ein Projekt dieses Umfangs zu gross wäre.

Der *File Manager* soll sich in Bezug auf Fehlervermeidung und -behandlung nach folgendem Modell verhalten:

- Direkte Benutzereingaben, -auswahlen und Programmfunktionen werden auf Korrektheit überprüft. Nach Außen hin kann dieses durch Engabeprüfungen und nachprüfende Dialoge, programmintern durch die Formulierung von Vor- und Nachbedingungen erfolgen. Falls hier ein Fehler auftritt wird der Benutzer darüber im entsprechenden Maße informiert. Beispiel für die Prüfung auf Korrektheit sind die Überprüfung von eingegebenen Dateinamen bei der Umbenennung einer Datei oder die interne Überprüfung, ob eine zu kopierende Datei auch in der programminternen Zwischenablage eingefügt worden ist.
- Wenn im Programm ein Fehler auftritt, soll der Benutzer mit einer aussagekräftigen Fehlermeldung informiert und ihm die Ursache des Fehler bekannt gegeben werden. Dies soll durch die Verwendung von Ausnahmen (*Exceptions*) realisiert werden.

Der Ablauf der Fehlertests kann dann nach dem nachfolgenden Fehlertestmodell aufgebaut werden:

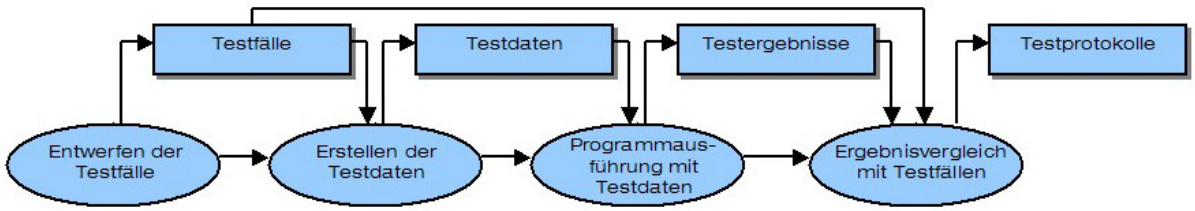

Abb. 6.1 Ablauf von Fehlertests

7 Dokumentationsanforderungen

Mit dem fertiggestellten Programm soll eine Dokumentation ausgeliefert werden, die folgende Elemente enthalten soll:

- *Vollständiger Quellcode*
  Der Quellcode der ausgelieferten Version wird mitgeliefert, so dass ein projektfremder interessierter Entwickler vollständig Einsicht in das System nehmen kann. Somit wird die Anpassung an individuelle Bedürfnissse ermöglicht.
  Die Rechte auf den ursprünglichen Quellcode liegen allerdings weiterhin bei dem tatsächlichen Systementwickler.

- *Anforderungsdefinition*
  Diese Anforderungsdefinition gibt die Anforderungen an den *File Manager* wieder, bevor dieser tatsächlich entworfen und implementiert wird. Sie ist somit ein wichtiger Bestandteil des Entwicklungsprozesses und soll zukünftigen Softwareentwicklern als Beispiel und Studienobjekt dienen.

- *Entwurf*
  Der Entwurf beschreibt die Konzeption und Planung des *File Managers*, bevor dieser implementiert wird. In ihm wird das Verhalten, das Design und die Funktionalität des Programms festgelegt. Auch der Entwurf ist ein wichtiger Bestandteil des Entwicklungsprozesses und kann ebenfalls als Beispiel und Studienobjekt dienen.

- *Benutzerhandbuch*
  Das Benutzerhandbuch beschreibt die Bedienung und die Funktionen des Programms. Es soll sowohl den Nutzer als auch den Administrator verständlich in die Bedienung einführen. Dazu werden die Funktionalitäten und die Leistungsfähigkeit des Programmes ausführlich beschrieben und erklärt.

- *Testdokumente*
  In den Testdokumenten ist aufgezeichnet, wie und mit welchen Mitteln das System getestet worden ist und welche Testdaten und -fälle währenddessen betrachtet worden sind. Zugriffsverletzungen und Systemausfälle werden genau betrachtet und analysiert. Eventuell noch verbliebene Fehler (Bugs) sollen mit dem Hinweis auf Bearbeitung angegeben sein.

- *JavaDoc-Dokumentation*
  Die *JavaDoc*-Dokumentation stellt eine genaue Dokumentation der Systemstruktur auf Quellcode-Ebene dar. Einzelne Klassen und ihre angebotenen Leistungen werden beschrieben, so dass ein projektfremder Entwickler ausreichend Einsicht zur eigenen Systemanpassung erhält.

Der Quellcode wird in Form von JAVA-Dateien vorliegen. Alle anderen Dokumente werden als PDF-Dateien ausgeliefert.

8 Akzeptanzkriterien

Die wichtigsten Akzeptanzkriterien für den *File Manager* sind die umgesetzten Funktionalitäten und die benutzerfreundliche Gestaltung der GUI. Wesentlich ist auch das Design des Entwurfes sowie die Beschreibung des gesamten Entwicklungsprozesses des *File Managers*.

Die Benutzeroberfläche ist die direkte Schnittstelle zum Anwender, und daher ist zum effektiven Einsatz des Systems und zur Erzielung hoher Anwenderakzeptanz wichtig, dass die GUI funktional und ansprechend gestaltet ist.

Die Beschreibung des Designs und des Entwicklungsprozesses des *File Managers* ist insofern wichtig, dass diese als Bestandteil eines Anwendungsbeispiels für die Ausbildung im Software Engineering verwendet werden soll. Dieses wird durch die mitgelieferte Dokumentation einschließlich Quellcode erfolgen.

Die Überprüfung der Kriterien erfolgt in einem Abnahmetest oder einer Projektbewertung, welche erst mit erfolgreichem Abschluss das System akzeptiert. Bei Nichteinhaltung der Anforderungen wird eine Konventionalstrafe in Form geminderter Bewertung gesetzt.

9 Glossar

- *Administrator*: Verwalter eines Softwaresystems oder eines Softwareprogrammes.

- *Algorithmus*: Eine zur Lösung eines bestimmten Problemes oder einer bestimmten Problemklasse mögliche Vorgehensweise/-folge.

- *Anforderungen*: Ansprüche, denen das zu entwickelnde Softwaresystem entsprechen muss. Die A. an ein Softwaresystem werden in der Anforderungsdefinition festgehalten.

- *Anforderungsdefinition*: Bei oder vor der Softwareentwicklung entstehendes Dokument, in dem funktionale, nichtfunktionale und generelle Anforderungen an die zu entwickelnde Software festgehalten werden. Die A. stellt oftmals die Vertragsgrundlage bei Entwicklungsaufträgen von Softwaresystemen dar.

- *Anwendungsfall*: Beschreibung eines speziellen Benutzungsszenarios der zu entwickelnden Software.

- *Anwendungsfalldiagramm*: Diagrammform in der UML, in der Anwendungsfälle der zu entwickelnden Software dargestellt sind.

- *Benutzer*: auch Nutzer oder User; der (End-)Benutzer eines Softwaresystemes.

- *Bibliotheken*: Sammlung von Programmen/Funktionen, die gebrauchsfertig bei Programmierung eines Programms zur Verfügung stehen (können), wenn sie im Lieferumfang eines Compilers enthalten sind. B. werden je nach Bedarf durch einen Aufruf in Programmsegmente eingebunden.

- *Browser*: z.B. Navigator von Netscape, Internet Explorer von Microsoft, Mosaic... Zeigt die von einem Server bereitgestellten Dokumente (meist in HTML) an. Er dient auch als Plattform, um Applets auszuführen.

- *Bug*: Fehler oder Absturz verursachender Quelltext in einem Software-System.

- *Dateiverwaltungsprogramm*: siehe File Manager.

- *Datenbank* (*DB* oder *DataBase*): System zur Speicherung und Abfrage von Daten.

- *Eclipse*: Open Source Software Entwicklungsumgebung; insbesondere ausgelegt für die Softwareentwicklung in Java.

- *EclipseUML*: Plugin für Eclipse zur Modellierung eines Softaresystems mit Mitteln der UML

- *EDV*: Elektronische Datenverarbeitung.

- *Exception*: Ausnahmefehler der während der Ausführung des Programmes (Laufzeitfehler; Stromausfall etc.) auftritt.

- *Fenster* (Ansicht): Ein Teil der grafischen Benutzeroberfläche. Ein Fenster ist ein Ausschnitt aus dem Bildschirm, in dem separat verschiedene Funktionen ablaufen können.

- *File Manager* (auch Dateiverwaltungsprogramm): Programm, dass den Zugriff auf ein Dateisystem verwaltet sowie das Öffnen, Schließen, Lesen und Schreiben von Dateien ermöglicht.

- *Funktionale Anforderungen*: Anforderungen, denen das Programm zur Erfüllung der von ihm zu leistenden Funktionalitäten genügen muss.

- *GUI = Graphical User Interface*: Die grafische Benutzeroberfläche; Schnittstelle zwischen dem Benutzer und der Funktionalität des Systems.

- *Hardware*: Die physischen Geräte eines Computers (zum Beispiel Drucker, Monitor usw.).

- *Input*: Eingegebene Daten (Text, Grafik, usw).

- *Intuitive Bedienung*: Das Programm ist ohne Vorkenntnisse bedienbar - man kann die Wirkung oder Vorgehensweise erahnen.

- *Java*: Programmiersprache von *Sun Microsystems*; JAVA wird besonders wegen seiner Plattformunabhängigkeit und seiner Netzwerkmöglichkeiten eingesetzt (Applets).

- *JavaDoc*: JDK-eigenes Tool zur Erstellung von Dokumentationen zu einem bestimmten Java-Programmquelltext.

- *JDK* = Java Development Kit: Von *Sun Microsystems* bereitgestellte Entwicklungsumgebung für JAVA; derzeit in der Version 1.4.2 bzw. 1.5 (beta) verfügbar.

- *JUnit*: Software zum Testen von Software.

- *Klasse*: Eine abstrakte Einheit eines Programms; besteht aus gleichartigen Objekten.

- *Konsistenz*: auch Systemkonsistenz: Grad der Übereinstimmung zwischen Meßwerten von Testeinzelleistungen; Softwarekorrektheit.

- *Konsistenzbedingungen*: Bedingungen, die zur Gewährleistung der Korrektheit von Software erfüllt sein müssen.

- *Menü*: Teil der grafischen Benutzeroberfläche; Befehle werden in Klartext auf dem Bildschirm ausgegeben und können über die Tastatur oder die Maus ausgewählt werden.

- *Online-Hilfe*: Ein Teil des Programms, das Funktionen des Programms während dessen Ablauf erklärt.

- *Package* (auch *Paket*): Pakete sind Ansammlungen von Modellelementen beliebigen Typs, mit denen das Gesamtmodell in kleinere überschaubare Einheiten gegliedert wird. Ein Paket definiert einen Namensraum, d.h. innerhalb eines Paketes müssen die Namen der enthaltenen Elemente eindeutig sein. Jedes Modellelement kann in anderen Paketen referenziert werden, gehört aber zu genau einem (Heimat-) Paket. Pakete können wiederum Pakete beinhalten. Das oberste Paket beinhaltet das gesamte System.

- *Plattformübergreifende Sprache*: Bezeichnung einer Sprache wie z.B. JAVA, welche auf jedem Computer lauffähig ist, ohne dass sie speziell angepasste Programme (Compiler) für das jeweilige Betriebssystem benötigt. So sind auf jedem Computer, auf dem eine Java-Virtual-Maschine installiert ist, Javaprogramme lauffähig; (unabhängig vom Betriebssystem).

- *Schnittstelle* (auch *Interface*); bezeichnet eine Struktur, welche zwischen verschiedenen 'Umgebungen' eine Kommunikation ermöglicht. (z.B. zwischen globaler und lokaler Umgebung, Nutzer und Programm).

- *Software*: Programme, die auf einem Computer ausgeführt werden können.

- *Softwaresystem*: System, dessen Systemkomponenten und -elemente aus Software bestehen; S. sind Produkte von Softwareprojekten.

- *Quellcode*: der Programmquelltext, in dem das Programm für eine bestimmte Programmiersprache lesbar entworfen worden ist.

- *Systemebene*: Umgebung innerhalb eines Systems, die sich von anderen (Sub-)Umgebungen desselben Systems abgrenzt.

- *Tool* (auch *CASE-Tool*): Werkzeugprogramm, welches die einfachere Bearbeitung komplexer Arbeitsvorgänge ermöglicht; oft auf visuellem Weg.

- *UML* = Unified Modelling Language: Grafische Sprache zur Beschreibung von Objekten eines Programms und deren Interaktionen.