Fachhochschule
Braunschweig/Wolfenbüttel
– University of Applied Sciences –

Fachbereich Informatik
Studiengang Praktische Informatik

Diplomarbeit

Erstellung eines statistischen Moduls für die systemüber-greifende Diagnose

(mit Realisierung der Spezialisierung SIMIS LC)

von

Erika Hernández Barrios
Matrikelnummer: 20166504

Aufgabenerstellung und Betreuung

Prof. Dr. F. Klawonn (Fachhochschule Braunschweig / Wolfenbüttel)
Dipl. -Ing. S. Waida (Siemens AG)

Braunschweig, 18.April 2005

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt habe.

Braunschweig, 18.April 2005 Unterschrift

Kurzfassung

(Deutsch)

Bei den SIMIS-Systemen werden Informationen in ein Diagnosemodul übertragen. Diese Informationen enthalten unter anderem Fehler und Störungsmeldungen, die zur weiteren Verarbeitung auf einen Service PC geladen werden.

Diese Diplomarbeit behandelt die Implementierung eines Statistik-Moduls für die Bahnübergangstechnik SIMIS LC, das das Lesen der Fehlerinformationen aus dem Diagnosemodul durchführt. Außerdem werden die Fehlerinformationen auf einem Diagnosetool ausgewertet und dargestellt. Das neue Modul wird an die von der Siemens AG entwickelnde "systemübergreifende Diagnose" angehängt. Für die Umsetzung soll mit einem objektorientierten Konzept gearbeitet werden.

Das Diagnosemodul unterstützt die schnelle Erkennung von Schwachstellen in den SIMIS-Systemen. Damit können Fehler und Ausfälle des Systems im Vorfeld erkannt und vermieden werden.

Abstract

(Englisch)

At the SIMIS-Systems the information will be send to the diagnostic module. These information withheld for example failures and error-signals, which are downloaded on a Service PC to be processed.

This thesis includes the implementation of a statistic module for the level crossing technique SIMIS LC, which executes the reading of the failure information out of the diagnostic module. Besides the failure information is processed and shown by the diagnostic tool. The module is directly connected with the "extended diagnostic system". For the transformation of the processed data it is needed to work with a object orientated Concept. The diagnostic module supports quick recognition of weaknesses in the SIMIS-Systems. Thereby failures and crashing of the system can be recognized in anticipation and can be prevented.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einführung

Statistik wird in den unterschiedlichen Bereichen der Wissenschaft als ein wertvolles Hilfsmittel eingesetzt. Sie basiert auf quantitativen Methoden und unterstützt in direkter Form das universelle Problem der intelligenten und richtigen Entscheidungsfindung. Statistik trägt dazu bei, große Datenmengen einfacher zu bearbeiten und zu bewerten.
Die von Siemens entwickelten SIMIS-Systeme (Sicheres Mikrocomputersysten von Siemens) senden Informationen, die in einem Diagnosemodul (DIMO) gespeichert werden. In den Informationen sind unter anderem Fehler und Störungsmeldungen enthalten. Diese Informationen sollen auf dem Diagnosemodul auf unterschiedliche Weise abgerufen werden können, z.B. über LAN oder Funkmodem. Da es verschiedene Systemtechniken (Bahnübergang, Achszählung, Gleisstromkreis, etc.) gibt, ist es unpraktisch, für jede Technik jeweils ein passendes Modul zu entwickeln. Aus diesem Grunde ist es sinnvoll, ein universelles Softwaremodul zur statistischen Fehlerdarstellung zu erstellen, welches auf die Architektur der bereits bestehenden „Systemübergreifenden Diagnose", einer universellen Diagnoseplattform, aufbaut. Das neue Softwaremodul soll fähig sein, Fehler von verschiedenen Quellen in einer Statistik darzustellen. Die gespeicherten Meldungen/Fehlerdateien auf dem Diagnosemodul sollen gelesen, ausgewertet und innerhalb der grafischen Oberfläche der "Systemübergreifenden Diagnose" angezeigt werden.
Dieses trägt dazu bei, eine effiziente Wartung der SIMIS-Systeme zu gewährleisten, und eine schnellere Identifizierung der Schwachstellen zu ermöglichen. Es können Fehler und Ausfälle des Systems rechtzeitig erkannt und vermieden werden.

Die vorliegende Diplomarbeit ist wie folgt aufgebaut:

Das Softwaremodul wird zunächst für die Bahnübergangstechnik SIMIS LC implementiert, daher wird in Kapitel Zwei eine Einführung in diese Technik dargestellt. In Kapitel Drei werden die verwendeten Konzepten der Statistik zusammengefasst. Die „Systemübergreifende Diagnose“ wird im Kapitel Vier kurz beschrieben.

Die Durchführung der Diplomarbeit geschah in 5 Phasen, wie es der bei Siemens verwendete Softwareentwicklungsprozesses PEACC (siehe Anhang 1) vorsieht.

Dazu sind im Kapitel Fünf zunächst die Anforderungen an das Modul detailliert aufgeführt, Architektur und Design werden im Kapitel Sechs beschrieben. Das Kapitel Sieben beschreibt die Implementierung des Moduls. Im Kapiteln Acht sind dann die Ergebnisse der Tests und der Validierungsphase dargestellt.

2. Die Bahnübergangstechnik SIMIS LC

SIMIS LC ist eine Bahnübergangstechnik der SIEMENS AG.

Unter einem Bahnübergang oder einer Eisenbahnkreuzung versteht man eine(Siehe Quelle (17))

„Höhengleiche Kreuzung einer Schienenbahn und einer Straße“

Da ein Zug nicht wie ein Straßenfahrzeug bei einer Kreuzung bremsen kann, muss daher das Kraftfahrzeug beim Herannahen eines Zuges anhalten. Die Bahnübergangsicherung ist die Maßnahme dazu, die verhindern soll, dass Straßenbenutzer einen Bahnübergang befahren bzw. betreten, wenn sich ein Eisenbahnfahrzeug nähert.

Es wird unterschieden zwischen Bahnübergängen mit technischer Sicherung und Bahnübergängen(Quelle (7)) ohne technische Sicherung.

Bahnübergang mit technischer Sicherung, bei dem durch straßenseitige Signal- und/oder Sperreinrichtungen (Lichtzeichen, Halbschranken, Schranken) das Befahren bzw. Betreten durch Straßenbenutzer unterbunden werden soll, wenn sich ein Eisenbahnfahrzeug nähert.

Bahnübergang ohne technische Sicherung, der nur durch Übersicht über die Bahnstrecke in Verbindung mit hörbaren Signalen der Eisenbahnfahrzeuge gesichert wird. Normalerweise ist hier nur ein Andreaskreuz vorhanden.
Zu einem Bahnübergang mit technischer Sicherung gehört eine Bahnübergangssicherungsanlage, die wärterbedient, zugesteuert oder signalgesteuert ausgeführt sein kann.
Die von der Siemens AG angebotenen Bahnübergangssicherungsanlagen bieten die Möglichkeit einer technischen Sicherung durch Überwachungssignale und durch Fernüberwachung.

Gängige Techniktypen sind zum Beispiel BUE 95F und SIMIS LC (Quelle (8)). Diese Techniken haben den Vorteil niedriger Wartungs- und Instandsetzungskosten bedingt durch eine optimale Diagnose über einen Service PC.

2.1 Beschreibung

SIMIS LC steht für Sicheres Mikrocomputersystem von Siemens Level Crossing und ist ein hochverfügbares, modulares Bahnübergangsicherungssystem, das für Fern-, Regional-, Industrie- und Stadtbahnen konzipiert ist (Siehe Quelle (8)).
Grundlage für SIMIS LC bilden die CENELEC-Normen (Siehe Quelle 15) für Eisenbahnanwendungen. Die in der Risikoanalyse für elektronische Bahnübergange ermittelten Sicherheitsziele werden erfüllt.

2.2 Aufbau

SIMIS LC besteht grundsätzlich aus der Außenanlage und dem Rechner SIMIS ECC (Element Control Computer), wie in Abbildung 1 gezeigt ist.

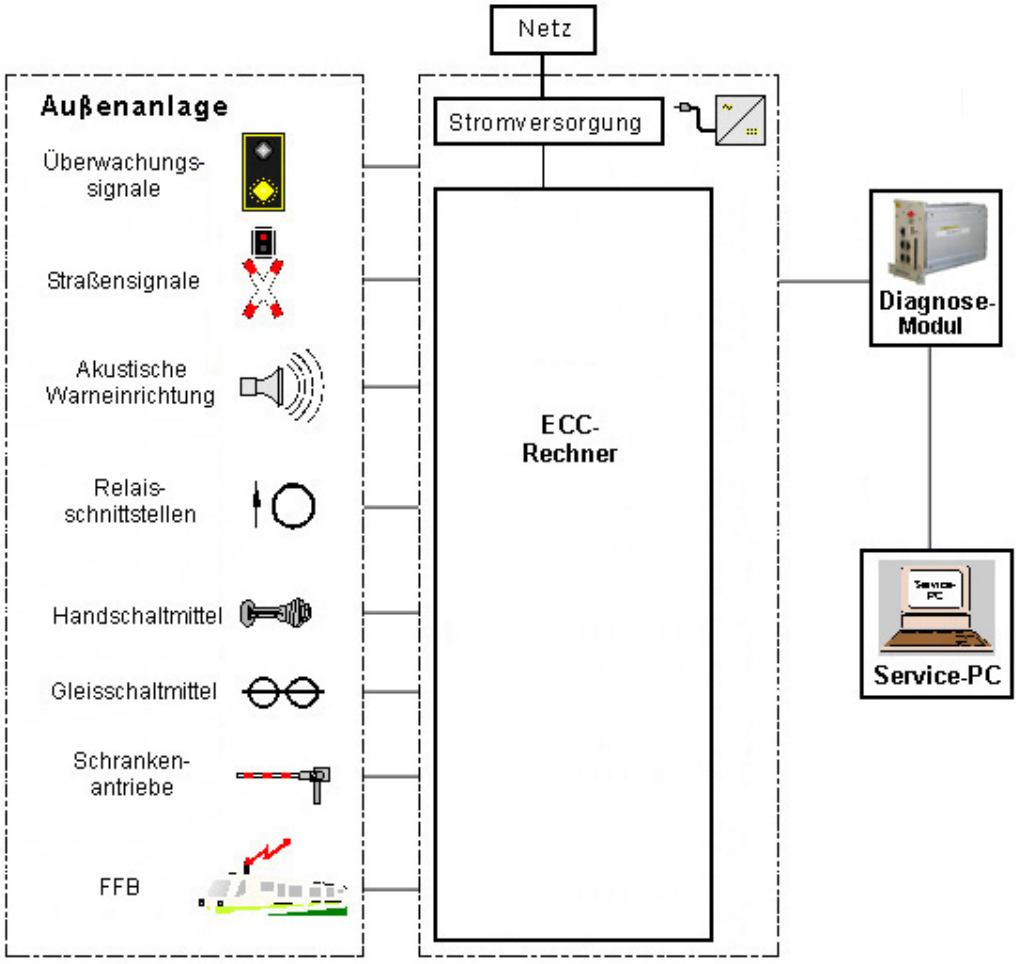


Abbildung 2.1 Übersicht zu SIMIS LC

Außenanlage

Die Außenanlage für SIMIS LC besteht aus folgenden möglichen Komponenten: Überwachungssignale, Straßensignale, Akustik-Einrichtungen, Relaisschnittstellen, Handschaltmittel, Schrankenantriebe und Übertragungssystem. Die Anzahl dieser Komponenten ist aber nicht festgelegt, was bedeutet, dass der Bahnübergang entsprechend örtlichen Gegebenheiten oder Kundenwünschen angepasst werden kann.

ECC-Rechner

Der sichere Rechner SIMIS ECC (Element Control Computer) ist die Standard-Hardware zur Steuerung und Überwachung der Komponenten der Außenanlage des Bahnübergangsicherungssystemen SIMIS LC und wird in der Regel als 2 von 3 System verwendet.

2.3 Diagnose-Modul

Das DIMO (Diagnose-Modul) ist ein eigenständiger Rechner, mit dem die Ferndiagnose für Bahnübergangstechnik SIMIS LC durchgeführt wird. Das Diagnosemodul ist in der Lage, über die Unilink-Schnittstelle Daten von bis zu 3 SIMIS-Rechnern simultan zu erfassen.

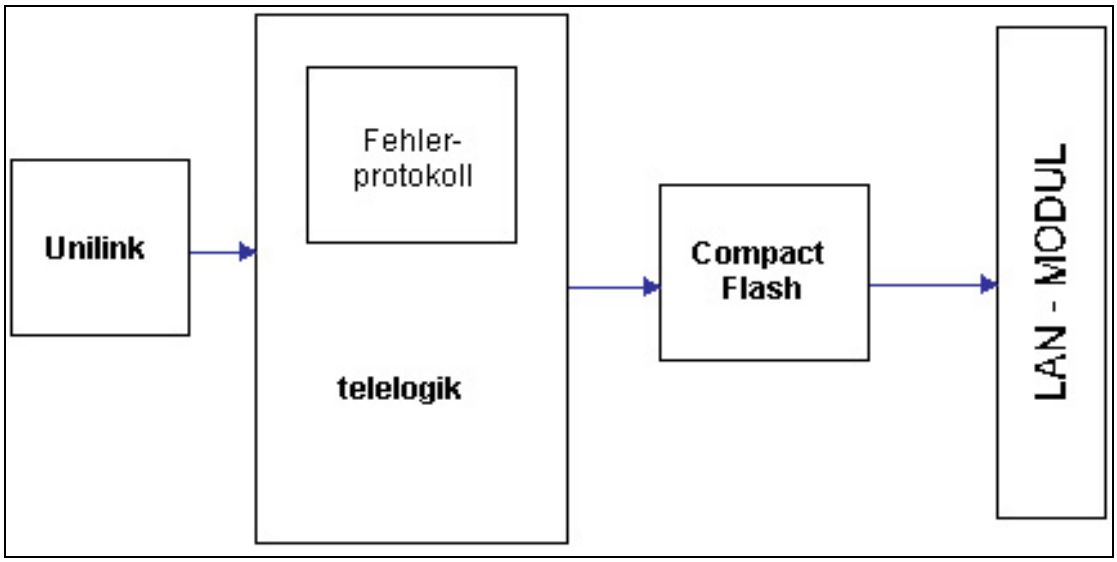


Abbildung 2.2 Fehlerspeicherung in DIMO

Die ECC-Rechner senden über die Unilink-Schnittstelle Informationen im Telegrammformat, die von dem Diagnosemodul empfangen werden. Die Telegramme enthalten unter anderem Daten aus Fehler- und Störungsmeldungen. Die **Telelogik** wertet diese Daten aus und speichert sie in einem **Fehlerprotokoll**-Speicherbereich. Dieser im RAM des DIMO angelegte Datenspeicher wird zyklisch auf das **Compact Flash** übertragen. Dann stehen die Daten als Fehlerdatei bereit, um sie z.B. mittels FTP über das **LAN-Modu**l zu lesen. Dieser Ablauf wird in der Abbildung 2.2 dargestellt. Nach jedem RESET der DIMO-Baugruppe wird eine neue Fehlerdatei erzeugt und abgespeichert. Der Dateiname setzt sich wie folgt zusammen:

Dateiname: fehlerlogYYMMDD_HHMMSS.txt

YY	Jahr
MM	Monat
DD	Tag
HH	Stunde
MM	Minute
SS	Sekunde

Die Datei enthält Informationen wie Meldung, Zeitpunkt und Zustand, die für das Statistik-Modul gebraucht werden.
Das **LAN-Modul** bildet die Schnittstelle vom DIMO zu einem externen Service-PC. Die Visualisierung der Daten und Steuerung des Diagnosemoduls erfolgt mittels Diagnose-Tools.

2.4 Service-Pc

Das Diagnose-Tool, welches auf einem Service-PC ausgeführt wird, dient zum:

- zyklischen Abrufen und Transferieren von lokal in dem DIMO aufgezeichneten Daten über ein Modem, LAN, IrDA, usw.
- Verarbeiten und Aufbereiten der Daten,
- Anzeigen von Meldungen (Betriebsmeldungen, Störungen, Fehlerzustände, Alarmierungen),

- Berechnen und Anzeigen von Wartungs- und Statistikdaten.

2.5 Fernüberwachung der Anlagen (Fü-Anlagen) in SIMIS LC

Die Fernüberwachung erfolgt durch das Diagnose-Tool auf dem Service-PC. Die im DIMO gespeicherten Fehlermeldungen entsprechen dem Zustand der In- und Außenanlage der Bahnübergangstechnik SIMIS LC. Bei den Fehlermeldungen wird unterschieden zwischen:

- Diagnose, Information
- Hinweise auf eine Fehlbedienung
- Fehler
- Störungen
- Alarmierungen

Zudem gibt es folgende Meldungstypen:

- Anlage ist ein- / ausgeschaltet,
- Anlage ist gesichert,
- Betrieb läuft dennoch weiter, sollte bald behoben werden,
- betriebshemmend, kein Normalbetrieb möglich,
- Zeitüberschreitung,
- Ansprechen der Ladefernüberwachung (Lfü).

3. Statistische Informationsdarstellung

3.1 Einleitung

Statistik ist die Wissenschaft und Kunst der Erfassung und Analyse von Datenstrukturen unter Berücksichtigung der unvermeidlichen Unschärfe, die durch zufällige Schwankungen und Fehler verursacht wird(Quelle 13).

Als **Statistik** wird bezeichnet:

1. **namengebend** (von *Status* »Staat«) die (vergleichende) Staatsbeschreibung (eingeführt wurde der Begriff vom Göttinger Kameralisten Gottfried Achenwall um 1749);
2. die heute als amtliche Statistik fortlebt;
3. davon verallgemeinernd Erhebungen aller Art, wie für Markt- und Meinungsforschung;
4. deren Ergebnisse, deren Darstellung u. a. die deskriptive Statistik betreibt;
5. die mathematische Statistik;
6. gewisse Zufallsvariablen, z.B. eine »Teststatistik«;
7. gewisse Modelle der statistischen Physik: Maxwell-Boltzmann-Satistik, Fermi-Dirac-Statistik, Bose-Einstein-Statistik.

3.2 Deskriptive Statistik

Die **deskriptive** oder **beschreibende Statistik** ist der älteste Zweig der Statistik, der sich vorwiegend mit der übersichtlichen Darstellung großer Datenmengen beschäftigt. Im Einzelnen gibt es folgende Möglichkeiten der Zusammenfassung:

- Tabellarische Auflistung
- Grafische Darstellung
- Berechnung von statistischen Kennwerten

Dies bildet das Instrumentarium, um umfangreiche Datenmengen auf einige wesentliche Informationen oder Maßzahlen zu reduzieren.
Die deskriptive Statistik betrachtet eine vorliegende Menge von Objekten, von denen für jedes Objekt ein Merkmal oder mehrere Merkmale beobachtet oder gemessen wurden. Man kann z. B. auszählen, wie oft die einzelnen Merkmalausprägungen vorkommen oder wie oft dieses in ein bestimmtes Intervall fallen. Die graphische Darstellung der Ergebnisse führt zur empirischen Häufigkeitsverteilung der gegebenen Datenmenge. Diese Verteilung gibt das Datenmaterial in übersichtlich aufbereiteter Form wieder.
Eine zufällige Datenmenge aus einer Menge von beliebigen Objekten (Personen, Tieren, Gegenständen usw.), die sich durch ein oder mehrere gemeinsame Merkmale auszeichnen, heißt eine Stichprobe. Umfasst diese Teilmenge genau n Objekte, so spricht man von einer Stichprobe vom Umfang n. Die Grundgesamtheit ist die Menge aller möglichen oder denkbaren Objekte, die sich durch dir gleiche Grundcharakteristik auszeichnen, z. B. die Grundgesamtheit aller Wolfenbütteler Studenten.

3.3 Tabellarische Darstellung

Bei statistischen Erhebungen werden die Beobachtungsergebnisse üblicherweise in der Reihenfolge nacheinander in Formulare oder Hefte eingetragen, in der sie anfallen. Die dabei stehende Liste heißt das **Protokoll** oder die **Urliste**. Tabelle 1 ist eine solche Urliste von 15 Werten von Personengewicht.

120	70	71
40	90	54
89	210	67
70	100	82
56	67	83

Tabelle 1 Urliste. Gewicht von 15 zwanzigjährigen Männern.

3.4 Häufigkeitsverteilung

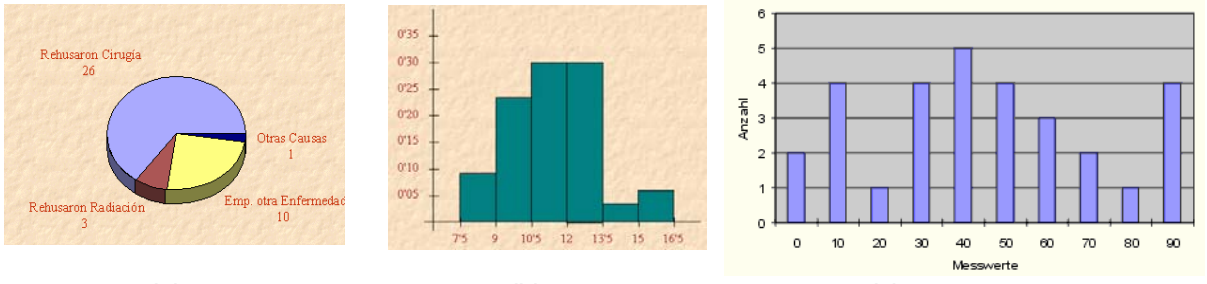
Die **Häufigkeit** gibt an, wie oft ein bestimmter Messwert auftritt. Um statistische Fragestellungen - welcher Art auch immer - zu untersuchen, werden Daten erhoben, die eine Quantifizierung erlauben. Gefundene Häufigkeiten sind also Grundlage bzw. "Herzstück" wissenschaftlicher Untersuchungsergebnisse.
Als **absolute Häufigkeit** bezeichnet man die Anzahl der Merkmalsträger, die einer bestimmten Messwertklasse zugeordnet sind. Die absolute Häufigkeit eines Messwertes stellt also dar, wie oft dieser bestimmte Messwert innerhalb einer Stichprobe vorkommt.

Eine **Häufigkeitsverteilung** ist eine Methode zur statistischen Beschreibung von Daten (Messwerten, Merkmalswerten). Mathematisch gesehen ist eine Häufigkeitsverteilung eine Funktion, die zu jedem Wert angibt, wie häufig dieser Wert vorgekommen ist. Man kann eine solche Verteilung als Tabelle, als Grafik oder modellhaft über eine Funktionsgleichung beschreiben (S. Quelle 10).

Graphische Darstellung

Es gibt verschiedene Diagramme zur graphischen Darstellung:

- Kreisdiagramm(Abb. 2, a)
- Stabdiagramm(Abb. 2 ,b)
- Histogramm(Abb. 2 ,c)



(a) (b) (c)

Abbildung 3.1 (a) Kreisdiagramm; (b) Stabdiagramm; (c) Histogramm.

4. Kurzbeschreibung des Diagnosesystem

Durch die bei Siemens entwickelte “Systemübergreifende Diagnose“ lassen sich verschiedener Bahntechniken diagnostizieren. Die Abbildung 4.1 zeigt die bisherige Struktur des Diagnosesystems.

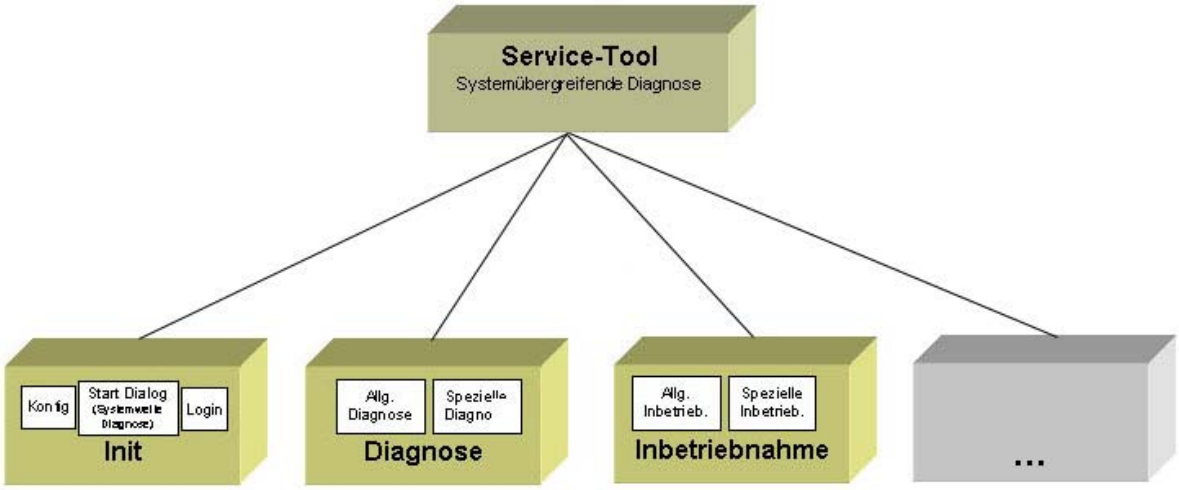


Abbildung 4.1 Übersicht der systemübergreifenden Diagnose

Das System teilt sich in drei Module. Das Modul „Init“ führt Aufgaben zur Initialisierung des Targets[1] durch. Die Parameter für die Schnittstellenkommunikation, sowie das verwendete Protokoll des Targets werden durch einen Startdialog definiert. Das Diagnosemodul erfasst und interpretiert die Ausgaben des gesamten Diagnosetargets bzw. einzelner Baugruppen. Das Inbetriebnahmemodul stellt Wartungsfunktionen zum Testen und Inbetriebnahmefunktionen verschiedener Anlagenkomponenten zur Verfügung.
Der letzte Teil der Abbildung bezeichnet die Erweiterungsmöglichkeit der systemweiten Diagnose. Hierher werden dann Module wie Konfiguration und Wartung eingesetzt.

Ziel der Arbeit, wie in der Einleitung erwähnt, ist die Erstellung eines statistischen Moduls, das in die „Systemübergreifende Diagnose“ eingebettet werden soll.

[1] Target bezeichnet einen Rechner von den SIMIS Systemen

5. Anforderungen

5.1 Einführung

In diesem Kapitel werden die Anforderungen an das statistische Modul beschrieben. Um in den folgenden Kapiteln auf die Anforderungen verweisen zu können, wird ein Indikator definiert, dem die folgende Struktur entspricht:

#REQ-Kategoriekürzel-Laufende.Nr**.#DEF#,**

wobei **REQ** Anforderung(*Requirement*) und **DEF** Definition entspricht.
Jeder Verweis einer Anforderung enthält **REF** (*Reference*) statt **DEF**.

Die Kategoriekürzel und ihre Entsprechungen:

FKT. Funktionale Anforderungen
DAR. Anforderungen an die Bedienung und Informationsdarstellung
DOK. Dokumentation
NFKT. Nicht funktionale Anforderungen

Die Tabelle 1 zeigt die entsprechenden Indikatoren je Kategorie.

Kategorie	Identifikator
Funktionale Anforderungen	#REQ-FKT-laufende.Nr#DEF#
Anforderungen an die Bedienung und Informationsdarstellung	#REQ-DAR-laufende.Nr#DEF#
Dokumentation	#REQ- DOK-laufende.Nr#DEF#
Nicht funktionale Anforderungen	#REQ-NFKT-laufende.Nr#DEF#

Tabelle 2 Kategorien von Anforderungen

5.2 Anforderungskatalog

Identifikator: #REQ-FKT-01#DEF#
Titel: Komponente Statistik
Beschreibung: Es wird ein Modul entwickelt, um die Fehler aus verschiedenen Systemtechniken statistisch darstellen zu können.

Identifikator: #REQ-FKT-02#DEF#
Titel: Fehlerdateien lesen
Beschreibung: Es wird eine Funktion definiert, die die entsprechende Fehler-Datei aus einem DIMO zur Verarbeitung lesen soll. Das Lesen der Datei kann auf verschiedene Weise erreicht werden. Entweder per FTP. oder in Telegrammform.

Identifikator: #REQ-FKT-03#DEF#
Titel: Zeitverlauf der Statistik
Beschreibung: Der zeitliche Verlauf der Fehler wird in der (Statistik-)Darstellung nur für ein gültiges Anfangsdatum eines Zeitraums berücksichtigt. Das bedeutet, es soll möglich sein, Statistiken für einen Tag; eine Woche;. einen Monat und einen Jahr darstellen zu können.

Identifikator: #REQ-DAR-01#DEF#
Titel: Informationsdarstellung
Beschreibung: Die Komponente Statistik soll eine graphische und dynamische Darstellung der Fehlermeldungen ermöglichen.

Identifikator: #REQ-DOK-01#DEF#
Titel: Dokumentation des Projektes
Beschreibung: Es wird ein offizielles Dokument geschrieben, wo die Software-Modul-Entwicklung beschrieben wird. Dieses Dokument wird auch als Diplomarbeit-Dokument dienen.

Identifikator:	#REQ-NFKT-01#DEF#
Titel:	Software-Erstellung
Beschreibung:	Es wird für die Software Erstellung Microsoft Visual Studio.NET (Version 2003). Verwendet und die Klassenbibliothek MFC(Microsoft Foundation Class Library) Version 7 genutzt

Identifikator:	#REQ-NFKT-02#DEF#
Titel:	Software-Technik
Beschreibung:	Das Modul wird objektorientiert programmiert.

Identifikator:	#REQ-NFKT-03#DEF#
Titel:	Software-Design
Beschreibung:	Es wird der CASE-Tool Rational Rose Professional C++ Edition Version 2003 zur Design der Klassenstruktur genutzt.

Identifikator:	#REQ-NFKT-04#DEF#
Titel:	Quellcode- und Dokumentenverwaltung
Beschreibung:	Es wird das Tool Rational Clear Case Explorer Version 2003 genutzt, um den Quellcode zu verwalten

6. Architektur und Design

In diesem Kapiteln wird die Einbettung des neuen Moduls in die Architektur der „Systemweiten Diagnose“ beschrieben. Zudem werden die Architektur und das Design des Statistikmoduls in verschiedenen Sichten, mit UML-Klassendiagrammen, Datenflussdiagrammen und einem Bedienoberflächeprototypen dargestellt.

6.1 Einbettung des Moduls

Die Abbildung 6.1 zeigt den Aufbau der „Systemweiten Diagnose“ mit ihren verschiedenen Modulen. In der Abbildung ist das Statistik Modul ein Bestandteil des Systems.

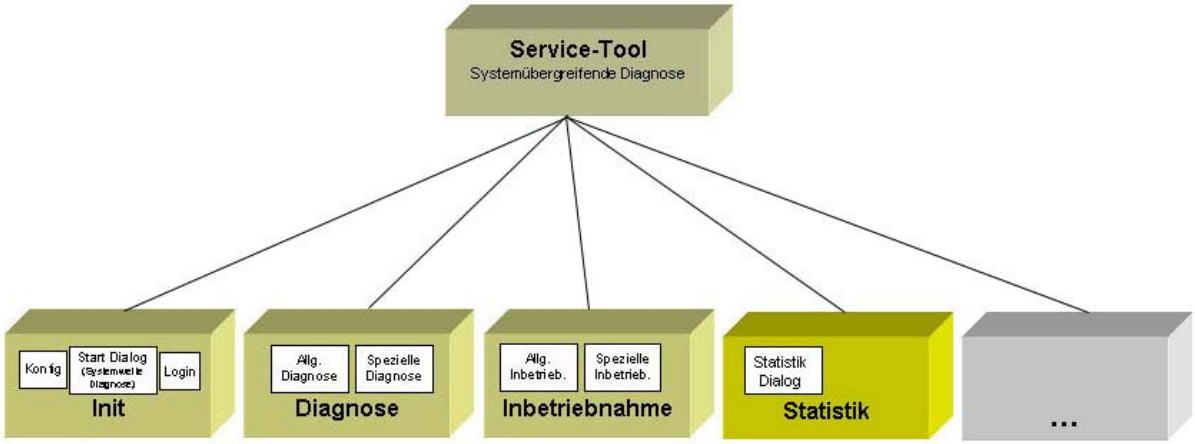


Abbildung 6.1 Übersicht der systemübergreifenden Diagnose

Dieses Statistikmodul besteht seinerseits aus Statistikobjekt mit drei Teilen: einem Kommunikationsobjekt, einem graphischen Objekt, und einem Darstellungsobjekt. Die Abbildung 6.2 zeigt das Statistik-Modul und seine Komponenten.

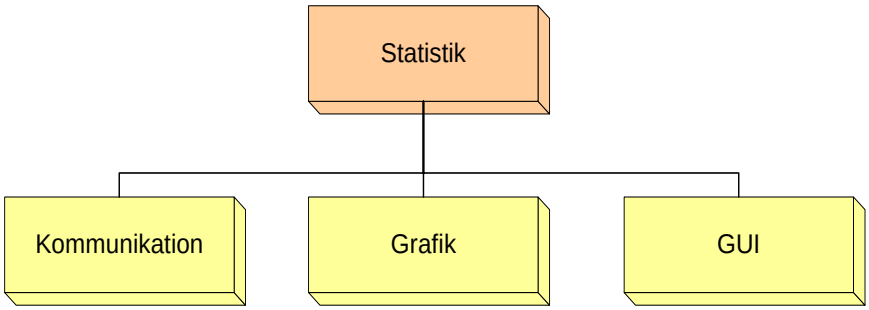


Abbildung 6.2 Struktur Statistik-Modul

Das Kommunikationsobjekt verbindet die systemübergreifende Diagnose mit dem Diagnosemodul. In ihr erfolgt das Auslesen von Fehlerdateien. Damit ist die Bearbeitung von Daten für die Statistik möglich. Abschließend ist die Darstellung auf einem GUI-Element von einem graphischen Objekt durchzuführen. Dies ist in der Abbildung 6.3 dargestellt.

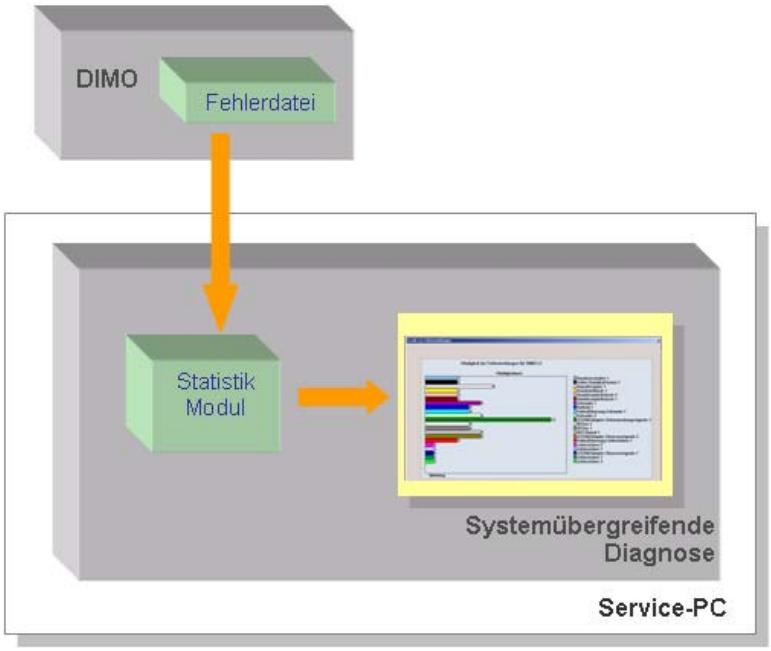


Abbildung 6.3 Funktionalität des statistischen Moduls

Datenflussdiagramm

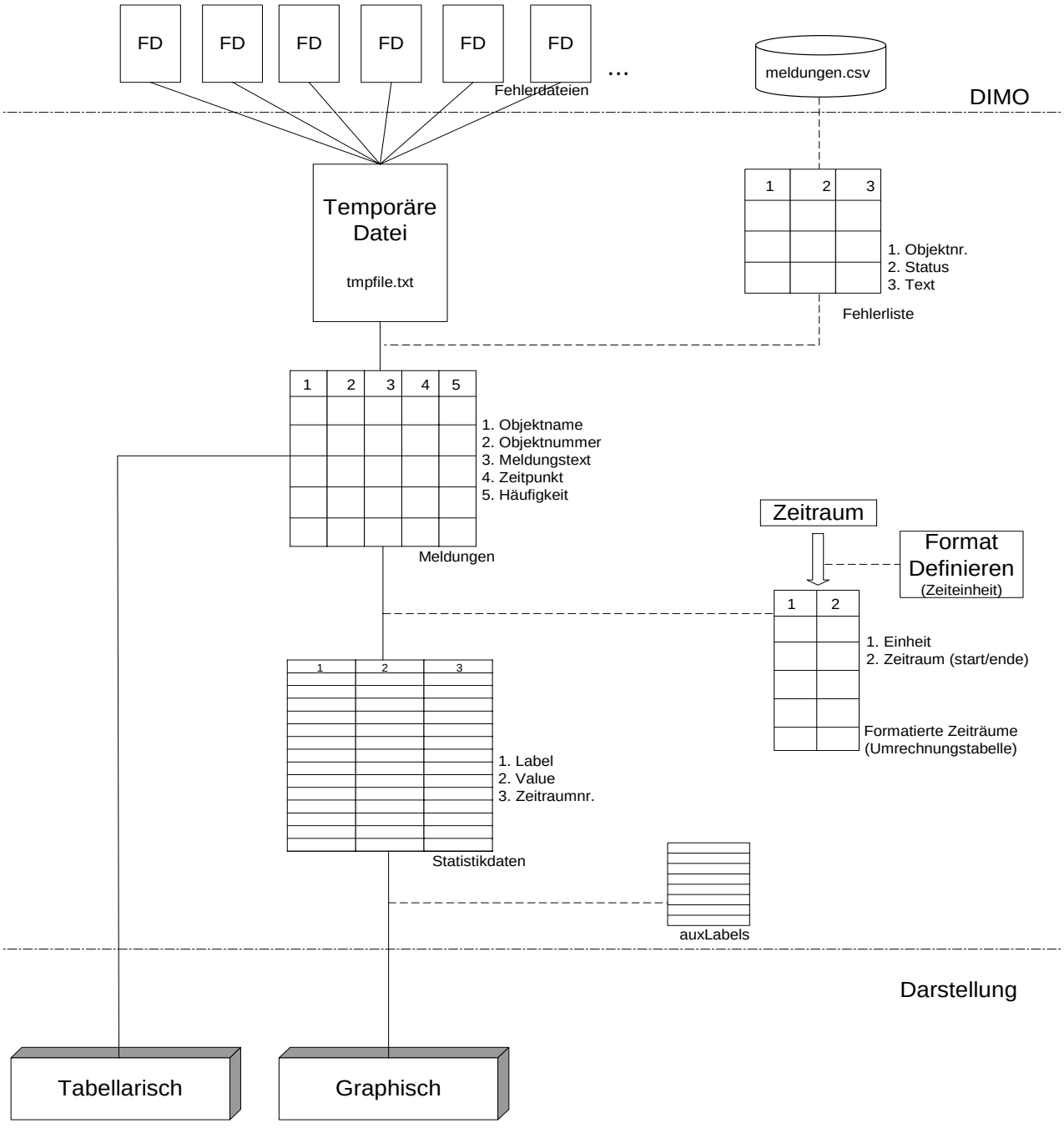


Abbildung 6.4 Datenfluss des statistischen Moduls

Die Abbildung 6.4 zeigt allgemein, wie die Datenbearbeitung in dem statistischen Modul durchgeführt wird.

Eine oder mehrere Fehlerdateien werden auf verschiedene Weise (z. B. beim der Bahnübergang über FTP) aus dem Diagnosemodul abgeholt und in einer gemeinsamen Datei lokal gespeichert. Danach werden mittels Vergleich mit der ebenfalls aus der DIMO gelesenen Datei „Meldung.csv“ identifiziert und in das Feld Meldung gelegt. Das Feld besteht aus 5 Spalten, die den Objektnamen, den entsprechendes Text, Objektnummer, Zeitpunkt in Sekunden und die Häufigkeit jedes Elementes enthalten. Für eine tabellarische Darstellung (siehe Beispiel im Anhang 2) von Fehlerinformationen wird dieses Feld benötigt.
Für eine graphische Darstellung wird zuerst der Zeitraum berücksichtigt. Eine Auswahl der Einheiten zur Unterteilung über den Zeitraum erfolgt über die Funktion „Format definieren“, die eine Zeiteinheit bestimmt. Damit wird das Feld Zeiträume ausgefüllt. Jedes Element dieses Feldes besteht aus zwei Zellen, eine mit dem Namen der Einheit und die andere mit dem schon formatierten Zeitraumstück, sodass auch der Start und das Ende gespeichert werden. Das bedeutet, das erste Element dieses Feldes entspricht dem Anfang des unformatierten Zeitraums und das letzte Element dem Ende. Diese Umrechnungstabelle dient zur Organisation und zur Sortierung der Daten aus dem Meldungsfeld. Nach der Umrechnung ist das Feld Statistikdaten bereit zur Visualisierung.

Kleinbeschreibung der Tabellen:

Fehlerliste. Diese Liste beinhaltet die Informationen zu jeder Objekt-ID, die das Modul Statistik zur Auswertung von Fehlerdateien benötigt. Die Informationen stammen aus der in dem Diagnosemodul gespeicherten Datei „Meldung.csv“. In folgendem Beispiel ist der Aufbau der Fehlerliste dargestellt:

Beispiel: Objekt: Lichtzeichen,
Objekt-ID=20,
Zustand 0 &1.

Objekt-ID	Zustand	Klartext
35	0	Lichtzeichen: Ausgeschaltet
35	1	Lichtzeichen: Ausgeschaltet, Hauptfaden defekt

Meldungen. Die Informationen, die aus den Fehlerdateien abgeholt werden, werden mittels der Fehlerliste ausgewertet und in der Tabelle Meldungen gespeichert. Das folgende Beispiel zeigt den Aufbau der Tabelle:

Beispiel: Zeitpunkt=1113217442 ,
Objekt_ID+Objektnummer=35001,
Zustand=1

Name	Text	Nummer	Zeitpunkt	Häufigkeit
Lichtzeichen	Ausgeschaltet, Hauptfaden defekt	1	1113217442	1

Zeiträume. Die Tabelle Zeiträume beinhaltet die Unterteilung eines Zeitraums in Abhängigkeit von einer Zeiteinheit. Diese Tabelle dient zur Umrechnung von der Information aus der Tabelle Meldungen. Ein Beispiel dafür ist folgendes:

Beispiel: Zeitraum: 10.04.2005-12.04.2005
Zeiteinheit: Tag

Einheit	Zeitraum(Start/Ende)	
Tag 1	10.04.2005	11.04.2005
Tag 2	11.04.2005	12.04.2005
Tag 3	12.04.2005	13.04.2005

Statistikdaten. Diese Tabelle, die auch als eine Spaltentabelle genannt sein könnte, beinhaltet umgerechnete Daten, die zu einer Statistik Bedeutung haben. Die Benutzung dieser Tabelle vereinfacht die Funktionen des Moduls zur graphischen Darstellung des Fehlerinformationen, wie folgendes Beispiel zeigt:

Beispiel: Meldung: Lichtzeichen wurde 20 Mal gefunden
Zeitpunkt: 11.04.2005

Label	Value(Häufigkeit)	NCol(Spaltennumer)
Lichtzeichen	20	2

Die Spalte 3, Spaltenummer ist von der Tabelle Zeiträume genommen und entspricht der Einheitsnummer von dem Zeitraum, in dem der Zeitpunkt der Meldung gefunden wurde.

auxLabels: Diese Liste hält die Reihenfolge von den Daten auf einer Grafik fest (auf einem bestimmten Achse). Jedes Labelfeld wird in dieser Liste ohne Wiederholung gespeichert. Wenn zum Beispiel die Meldung Lichtzeichen zuerst ausgewertet wird, wird ihr auch der erste Platz auf der Liste auxLabels und auf der Grafik zugewiesen.

6.2 Klassendiagramm

#REQ-NFKT-03#DEF#

Das Modul wurde nach den UML-Modellierungsrichtlinien erstellt. Zur Erstellung des Klassenmodels diente das CASE-Tool Rational Rose. Folgende Abbildung zeigt das allgemeine Klassenmodel des Moduls:

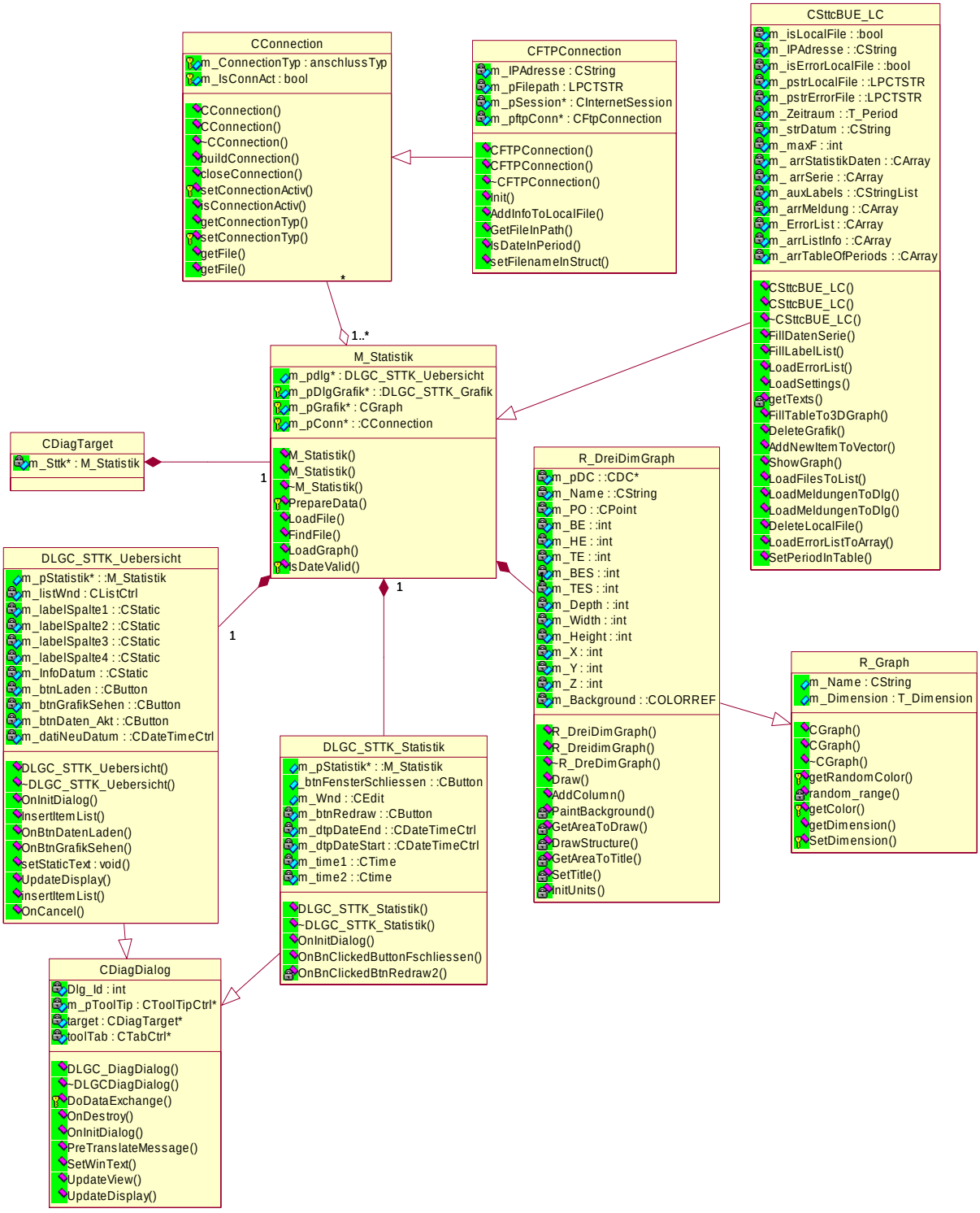


Abbildung 6.5 Klassendiagramm des Statistik Moduls

6.3 Klassenbeschreibung

6.3.1 Klasse „M_Statistik“

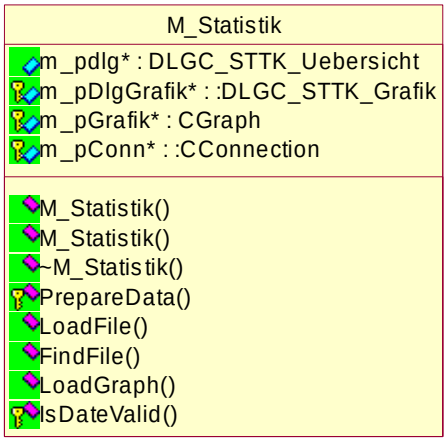


Abbildung 6.6 Basisklasse des statistischen Moduls

Klassename: `M_Statistik`. Wo „M“ bezeichnet Modul.

Beschreibung: Hauptklasse des Moduls, hier sind die Aufgaben zum Datenlesen, zur Bearbeitung und Konfiguration der Fehlerdateien zur statistische Bildung definiert. Diese Klasse stellt die Basisklasse für alle Techniken dar. Siehe Abbildung 6.3.

Zugriff: Public

Variablen:

Nicht öffentliche

`CConnection* m_pConn`	Zeiger auf die Verbindung(FTP; UNILINK; usw.), wodurch die Fehlerdatei abgeholt wird.
`DLGC_STTK_Statistik* m_pDlgGrafik`	Zeiger auf den Dialog, wo die graphische Darstellung ausgegeben wird.
`R_Graph* m_pGrafik`	Zeiger auf ein Grafik-Objekt.

Offentliche

`DLGC_STTK_Uebersicht* m_pDlg`	Zeiger auf den Dialog Übersicht zur tabellarischen Darstellung. Diese Variable wurde öffentlich definiert, weil der Dialog in der Methode zur Erstellung der Karteikarte von jedem Target erzeugt wird.

Methoden:

Öffentliche

`void M_Statistik(void);`	Standard Konstruktor der Klasse
`void M_Statistik(CString IPAdresse);`	Standard Konstruktor der Klasse. Er Bezeichnet, eine Verbindung via FTP von der Bahnübergangtechnik
`virtual ~M_Statistik(void);`	Falls die Zeiger des Moduls Daten enthalten, werden sie im Destruktor gelöscht und wieder auf NULL gelegt.
`bool LoadFile (CTime date1, CTime date2);`	Wertet die Daten aus einem bestimmten Zeitraum aus und führt Funktionen zur Darstellung auf einem Dialog aus
`bool LoadGraph(LPCTSTR Filename);`	Wertet die schon von PrepareData() vorbereiteten Daten aus und führt damit eine graphische Darstellung auf einem Dialog aus.

Nicht öffentliche Methoden

`bool PrepareData(void);`	Bereitet die schon abgeholten Fehlerdaten zur statistischen Auswertung vor.
`bool IsDateValid(CTime Date);`	Liefert eine positive Rückmeldung, wenn ein gegebenes Datum vorhanden ist.

Oberklasse: Keine

Unterklassen: CSttk_BUE_LC

Beziehungen: Einem Target gehört ein Statistik-Modul

Die klasse M_Statistik stellt die Basisklasse für alle möglichen Systemtechniken dar. Von ihr erben die technikspezifischen Klassen für die Bahnübergangtechnik (CSttk_BUE_LC), Gleisstromkreistechnik (CSttk_GSK_TCR), Achzählungsstechnik (CSTTK_AZ_S350U). Wie die klassen abgeleitet sind, wird ebenfalls den Datentyp durch den Name spezifiziert, z. B. CSttkBUE_LC enthaltet C aus Klasse, Sttk von Statistik, BUE aus Bahnübergang und LC von SIMIS LC. Die Abbildung 6.7 zeigt die Vererbung der Klasse M_Statistik.

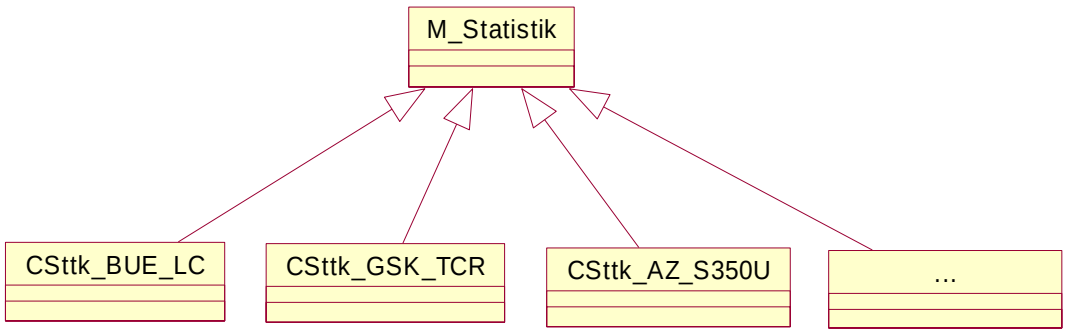


Abbildung 6.7 Vererbungsdiagramm der Statistik Klasse

6.3.2 Klasse „CSttkBUE_LC“

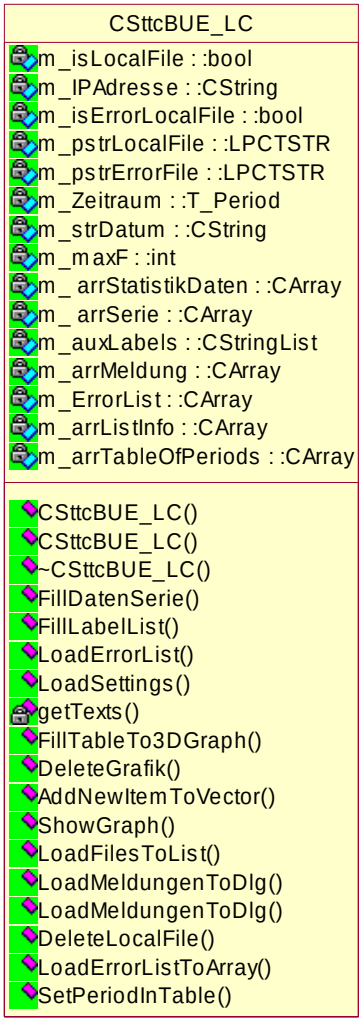


Abbildung 6.8 Spezielle Klasse für SIMIS LC

Klassename: `CSttkBUE_LC`.

Beschreibung: Diese Klasse stellt alle für die bei der Bahnübergangstechnik SIMIS LC benötigten statistischen Funktionen zur Verfügung.

Zugriff: Public

Variablen:

Private

`CString m_IPAdresse;`	IP-Adresse des DIMO’s
`T_Period m_Zeitraum;`	Bestimmter Zeitraum.
`CArray<T_Register, T_Register&> m_arrMeldung;`	Enthält die aus der Datei zu bearbeitenden Meldungen(Basisliste)

`CArray<T_ErrorList, T_Errorlist&> m_arrErrorList;`	Liste der technischen Fehlermeldungen aus der *.scv Fehlerdatei.
`CArray<T_Datenserie, T_Datenserie&> m_arrlistInfo;`	Zwischenspeicher für Meldungen und entsprechende Häufigkeitswerte
`CArray<T_InfoPeriod, T_Infoperiod&> m_arrTableOfPeriods;`	Enthält die Verteilung eines bestimmten Zeitraums in Tagen, Wochen, Monaten, Quartalen oder Jahren.
`CStringList m_auxLabels;`	Enthält die Etikettennamen, die für die graphische Darstellung nötig sind.
`CArray<T_StatistikDaten, T_StatisikDaten&> m_arrStatistikDaten;`	Alle Daten zur graphischen Darstellung sind vorbereitet und in dieser Reihe gespeichert.
`CArray<T_Datenserie, T_Datenserie&> m_arrSerie;`	Temporäre Serie von Meldungen und Häufigkeitswerten.
`bool m_islocalFile;`	Eine lokale Datei ist vorhanden.
`bool m_isErrorlocalFile;`	Es wurde eine Fehlerdatei lokal gespeichert.
`CString m_strDatum;`	Datum der aktuellsten Datei, die auf dem DIMO gefunden wurde.
`int m_maxF;`	Höchster Häufigkeitswert.
`LPCTSTR m_pstrLocalFile;`	Verzeichnis einer lokalen Datei zur temporären Nutzung.
`LPCTSTR m_pstrErrorFile;`	Verzeichnis einer Fehlermeldungsdatei.

Methoden:

Private

`CArray<T_Datenserie, T_Datenserie&>* FillDatenSerie( T_Period period);`	Wählt alle Objekte aus der Meldungsliste aus, die mit dem angegebenen Zeitraum übereinstimmen und füllt damit eine Liste mit Objektnamen und Häufigkeitswerten.
`CStringList* FillLabelList(void);`	Diese Methode füllt die Liste der Etiketten aus und liefert sie zurück.
`bool LoadFilesToList(CTime Start, CTime End);`	Lädt eine Liste mit den Namen aller vorhandenen Dateien in einem bestimmten Datum oder Zeitraum. Liefert *false*, wenn die Datei leer ist.
`CString GetTexts(int iObjectID, vint iStateCode);`	Vergleicht die Objekt-ID und den Zustandkode mit der Fehlerliste und liefert der entsprechenden Text der Fehler zurück. Liefert einen leere String, wenn kein Text gefunden wurde.
`bool LoadErrorListToArray(void);`	Liest Informationen zu jeder Objekt-ID und speichert sie in einem Feld(Array).
`void FillTableTo3DGraph(void);`	Bereitet eine mehrere dimensionale Tabelle mit ausgewerteten Daten zur graphische Darstellung.
`CArray<T_Datenserie, T_Datenserie&>* GetTable(void);`	Liefert einen Zeiger auf einer Serie von Meldungen.
`void DeleteGrafik(void) ;`	Löscht den Zeiger zur Grafik, wenn es nötig ist.
`INT_PTR AddNewItemToVektor(INT_PTR index, const CString &newObj, const CString &newErItem, int &inewTime, int newNr);`	Fügt ein neues Element in die Meldungliste ein.
`void ShowGraph(CDC *pdc);`	Zeichnet eine Grafik in den angegebenen Gerätkontext.
`bool LoadMeldungenToDlg(void);`	Lädt bestimmte Meldungen mit ihren Beschreibungen in eine Liste.

`bool LoadMeldungenToDlg(CTime tDate);`	Lädt Fehlermeldungen eines bestimmten Datums mit ihren Beschreibungen in eine Liste.
`void ShowMeldungen(void);`	Durch diese Funktion werden die schon vorbereiteten Daten in einem Dialog angezeigt.
`void DeleteLocalFile(LPCTSTR pstrLocalFile);`	Löscht existierende lokale Datei.
`bool LoadErrorListToArray(void);`	Lädt die Meldungen aus der Fehlerdatei.
`void SetPeriodInTable(T_Period Period, T_PeriodTyp RangeTyp);`	Füllt eine Tabelle mit der Verteilung des Zeitraumes aus.

Oberklasse: Keine

Unterklassen: CSttk_BUE

Beziehungen: Keine

6.3.3 Klasse „Connection“

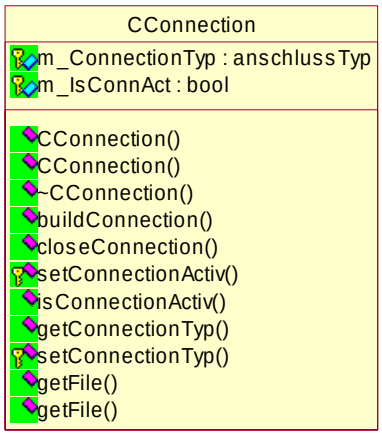


Abbildung 6.9 Klasse CConnection

Klassename: CConnection

Beschreibung: Diese Klasse stellt Funktionen zur Kommunikation über die unterschiedliche Schnittstelle des DIMO’s bereit.

Zugriff: Public

Variablen:

Nicht Öffentliche

`AnschlussTyp m_ConnecTyp;`	Aktive Verbindungsart
`bool IsConnAct;`	Verbindung ist Aktiv.

Methoden: Die meisten Methoden sind abstrakt definiert und müssen von jeder abgeleiteten Klasse überschrieben werden.

Öffentliche

`void CConnection(void);`	Standard Konstruktor
`void CConnection(CString IPAdresse);`	Standard Konstruktor für eine FTP-Verbindung
`virtual void ~CConnection(void);`	Standard Destruktor
`void BuildConnection(void);`	Erstellt Funktionen zum Verbindungsaufbau zur Verfügung.
`void CloseConnection(void);`	Enthält Funktionen zum Schließen der aufgebauten Verbindung
`void GetFile(void);`	Stellt Funktionen zur Dateisuche zur Verfügung.
`void GetFile(CTime start, CTime end);`	
`AnschlussTyp GetConnTyp(void);`	Liefert die Art der Verbindung zurück.
`bool IsConnActiv(void);`	Liefert den Zustand der Verbindung zurück.

Nicht Öffentliche

`void SetConnectionActiv(void);`	Speichert den Zustand der Verbindung in der Variable m_IsConnActiv.
`void SetConnectionTyp(AnschlussTyp typ);`	Speichert die Verbindungsart in der Variable m_ConnecTyp.

Oberklasse: Keine

Unterklassen: CFTPConnection

Beziehungen: Einem Statistik-Modul gehört ein CConnection-Objekt

Die Klasse CConnection stellt wie die Klasse M_Statistik die Basisklasse für die verschiedene Verbindungsmöglichkeiten mit dem Diagnosemodul dar. Die Abbildung 6.11 zeigt die mögliche Vererbungsdiagramm der CConnection Klasse.

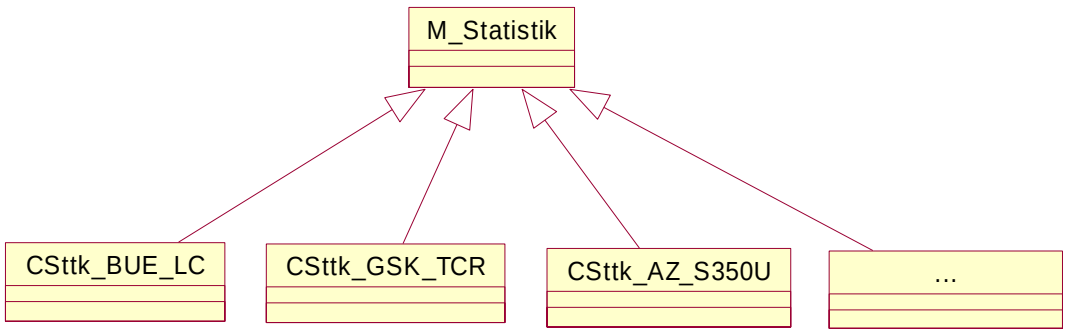


Abbildung 6.10 Vererbungsdiagramm der CConnection Klasse

6.3.4 Klasse „CFTPConnection“

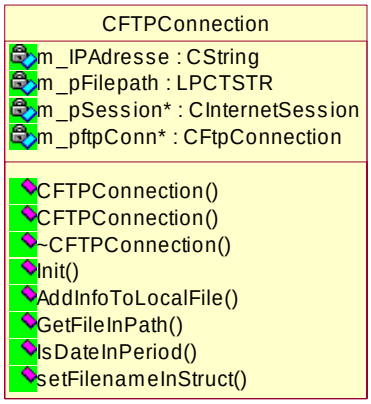


Abbildung 6.11 Spezielle Klasse für FTP-Verbindung

Klassename: CFTP_Connection

Beschreibung: Diese Klasse enthält alle Funktionen, um eine Verbindung via FTP-Protokoll aufzubauen.

Zugriff: Public

Variablen:

Nicht Öffentliche

`CString m_IPAdresse;`	IP-Adresse des DIMOS
`CinternetSession* m_pSession ;`	Zeiger auf ein Objekt Internetsitzung
`CFTPConnection* m_pFTPConn;`	Zeiger auf ein Objekt FTP-Verbindung
`LPCTSTR m_pFilepath;`	Verzeichnis auf dem DIMO, woher Dateien ausgelesen werden.

Methoden:

Nicht Öffentliche

`void CFTPConnection(void);`	Standard Konstruktor
`void CFTPConnection(CString IPAdresse);`	Standard Konstruktor für eine FTP-Verbindung
`virtual void ~CFTPConnection(void);`	Standard Destruktor
`void SetFilenameInStr(Cstring Filename);`	Zerlegt einen Dateiname in seinen einzelnen Elementen(Name, Tag, Monat, Jahr)
`Bool IsDateInPeriod(Ctime Date);`	Überprüft, ob das angegebene Datum, ein gültiges Datum ist.
`Bool GetFileInPath(LPCTSTR pstrFilename, LPCTSTR pstrPath);`	Setzt die Verbindung auf das angegebenen Verzeichnis und sucht die angegebene Datei.
`Void AddInfotolocalFile(Carray arrMeldungen);`	Speichert den Inhalt aller gefundenen Fehlerdateien in einer einzigen lokalen Datei.

Oberklasse: Cconnection

Unterklassen: Keine

Beziehungen: Keine

6.3.5 Klasse „Graph“

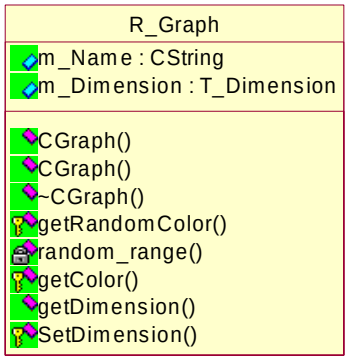


Abbildung 6.12 Klasse Graph

Klassename: R_Graph

Beschreibung: Diese Klasse erstellt die Basisfunktionalität, um ein Grafisches Objekt zu erzeugen. Der Name enthält „R“ aus Ressource.

Zugriff: Public

Variablen:

Nicht Öffentliche

`Cstring m_Name;`	Name der Grafik.
`T_Dimension;`	Dimension der Grafik.

Methoden:

Öffentliche

`void R_Graph(void);`	Standard Konstruktor.
`Virtual void R_Graph(void);`	Standard Destruktor.
`Bool GetFileInPath(LPCTSTR pstrFilename, LPCTSTR pstrPath);`	Setzt die Verbindung auf das angegebene Verzeichnis und sucht die angegebene Datei.
`Void AddInfotolocalFile(Carray arrMeldungen);`	Speichert den Inhalt aller gefundenen Fehlerdateien in einer einzigen lokalen Datei.

Oberklasse: Keine

Unterklassen: R_BarGraph, R_DreiDimGraph

Beziehungen: Einem Statistik-Modul gehört eine oder mehrere Grafiken

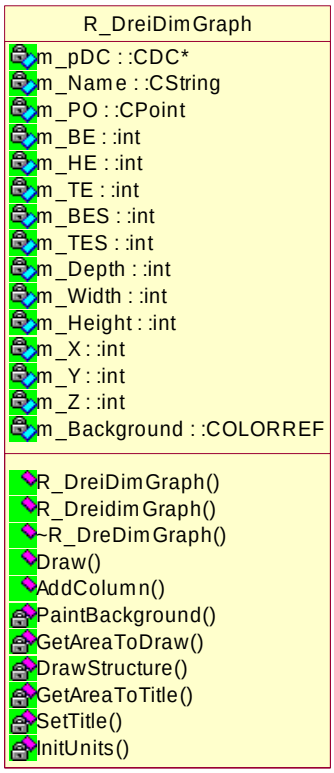


Abbildung 6.13 Klasse R_DreiDimGraph

Klassename: R_DreiDimGraph

Beschreibung: Diese Klasse bietet Funktionen zum Aufbau und Ausfüllung einer dreidimensionalen Grafik.

Zugriff: Public

Variablen:

Nicht Öffentliche

`CDC*  m_pDc;`	Verwaltung der Oberfläche zum Zeichnen
`Cstring m_Name;`	Name der Grafik
`Cpoint m_PO;`	Point Origin der Grafik
`int  m_BE;`	Breiteeinheit pro Spalte
`int  m_HE;`	Höheeinheit pro Spalte
`int  m_TE;`	Tiefeeinheit pro Spalte
`int  m_BES;`	Breiteeinheit Innerhalb jeder Spalte
`int  m_TES ;`	Höheeinheit Innerhalb jeder Spalte
`int  m_Depth;`	Tiefe (Total)
`int  m_Width;`	Breite (Total)
`int  m_Height;`	Höhe (Total)
`int  m_X, m_Y, m_Z;`	Größter Wert für jede Achse
`COLORREF m_Background;`	Hintergrundfarbe der Grafik

Methoden:

Öffentliche

`R_DreDimGraph(void);`	Standard Konstruktor.
`R_DreDimGraph(CDC *pdc, CString strName, int X_Value, int Y_Value, int Z_Value, COLORREF cBackground);`	Spezieller Konstruktor.
`~R_DreDimGraph(void);`	Standard Destruktor
`bool Draw(void);`	Zeichnet eine Grafik in Dreidimensionen mit den angegebenen Eigenschaften (Ohne Werte)
`void AddColumn( int x, int y, int z );`	Fügt eine neue Spalte an der entsprechenden Koordinate ein

Nicht Öffentliche

`void PaintBackground (CRect rArea, COLORREF cColor);`	Zeichnet den Hintergrund in hell blauer Farbe (default)
`CRect GetAreaToDraw (CRect rect );`	Liefert die x, y, x1, y1 Koordinate der Fläche zurück, wo die Grafik gezeichnet wird . Zuerst wird die gesamte Fläche in drei Teile geteilt. Die ersten beiden Teile werden als Oberfläche für die Grafik verwendet. Der dritte Teil wird für Kommentare reserviert.
`void DrawStructure( CPoint pOriginPoint, int iTopBorder );`	Zeichnet den Hintergrund der Grafik und organisiert die Verteilung der Reihen und Spalten
`CRect GetAreaToTitle( CRect rArea );`	Liefert die Koordinaten eines Rechteck zurück, wo der Grafiktitel gezeichnet wird. Das entspricht einem Zehntel des gesamten Bereichs (oben)
`void SetTitle( CRect rArea, CString strTitle );`	Setzt den Namen der Grafik für den Dialog
`void InitUnits( void );`	Initialisiert die Einheitswerte

Oberklasse: R_Graph

Unterklassen: Keine

Beziehungen: Keine

6.3.6 Klasse „CDiagDialog“

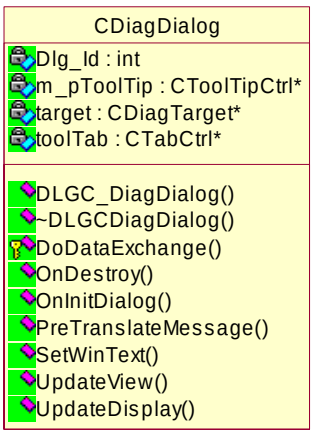


Abbildung 6.14 Klasse CDiagDialog

Klassename: CDiagdialog

Beschreibung: Diese Klasse beinhaltet die Basisfunktionen zu dem Dialogen der Diagnose.

Zugriff: Public

Variablen:

Öffentliche

`CDiagTarget* target;`	Zeiger auf das Target, zu dem dieser Dialog gehört.
`CTabCtrl* toolTab;`	Zeiger auf den benutzten ToolTab.
`enum { IDD = IDD_DLGC_DIAGDIALOG };` `int Dlg_Id;`	Dialogfelddaten. Identifiziert den Dialog. Hier steht die "IDD" des Dialogs.
`CToolTipCtrl* m_pToolTip;`	Zeiger auf den ToolTipCtrl.

Methoden:

Öffentliche

`DLGC_DiagDialog(UINT IDD,CWnd* pParent = NULL);`	Standardkonstruktor
`virtual ~DLGC_DiagDialog();`	Standard Destruktor.
`virtual bool UpDate-Display();`	Erneuert die angezeigten Werte.
`virtual void UpdateView();`	Aktualisiert den gesamten View.
`virtual void SetWinText(int nWndID, UINT nStringID);`	Setzt den Windows Text eines GUI Elementes. @para int nWndID ID des Elementes @para UINT StringID ID des Strings aus dem StringTable
`virtual BOOL PreTranslateMes-sage(MSG* pMsg);`	
`virtual BOOL OnInitDialog();`	
`afx_msg void OnDestroy();`	

Nicht Öffentliche

`virtual void DoDataEx-change(CDataExchange* pDX);`	DDX/DDV-Unterstützung

Oberklasse: CDiagDialog

Unterklassen: Keine

Beziehungen: Jedem statistischen Modul entspricht ein DLGC_Uebersicht Dialog

6.3.7 Klasse „DLG_Uebersicht“

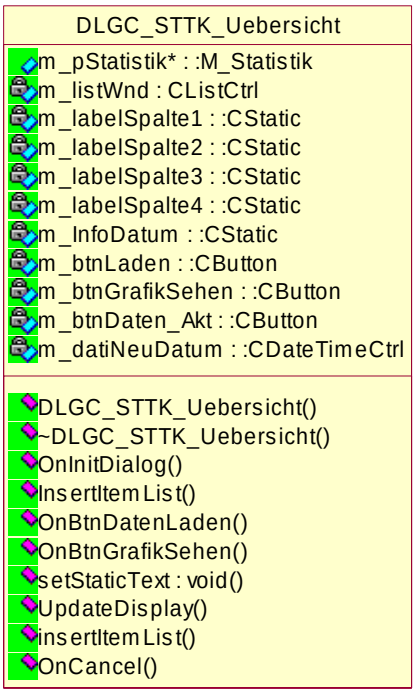


Abbildung 6.15 Klasse DLGC_STTK_Uebersicht

Klassename: DLG_Uebersicht

Beschreibung:

Zugriff: Public

Variablen:

Öffentliche

`M_Statistik*  pStatistik;`	Zeiger auf das Modul, woher die Daten stammen

Nicht Öffentliche

`CListCtrl m_listWnd;`	Liste auf dem Bildschirm
`CStatic m_LabelSpalte1;`	Objektname (Liste)
`CStatic m_LabelSpalte2;`	Objektnummer (Liste)
`CStatic m_LabelSpalte3;`	Fehlermeldung (Liste)
`CStatic m_LabelSpalte4;`	Zeitpunkte (Liste)
`CStatic m_Info_Datum;`	Etikett "Info zum"
`CButton m_btnLaden;`	Ruft Daten aus der Statistik
`CButton m_btnGrafikSehen;`	Ruft die Dialog-Grafik auf
`CButton m_btnDaten_Akt;`	Aktualisiert die Daten der Tabelle
`CDateTimeCtrl m_datiNeuDatum;`	Neues Datum

Methoden:

Öffentliche

`DLGC_STTK_Uebersicht(M_Statistik*`	Standard Konstruktor

`_pStatistik, CWnd* pParent = NULL);`	
`virtual ~DLGC_STTK_Uebersicht();`	Standard Destruktor
`virtual BOOL OnInitDialog();`	Initialisierung des Dialoges
`void insertItemList(int nItem, CString Item, CString & subItem1, CString & subItem2, CString & subItem3);`	Fügt eine neue Zeile in die Liste ein
`afx_msg void OnBnClickedButtonLaden();`	Fügt die schon bearbeiteten Daten zur Darstellung in einen Listcontrol.
`afx_msg void OnBnClickedButtonGrafikensehen();`	Dialog Statistik wird angezeigt.
Nicht Öffentliche	
`void setStaticText(void);`	Setzt die sprachabhängigen Texte.
`void OnCancel();`	Deaktiviert die Aktion Cancel

Oberklasse: CDiagDialog

Unterklassen: Keine

Beziehungen: Jedem statistischen Modul entspricht ein DLGC_Uebersicht Dialog

6.3.8 Klasse „DLG_STTK_Statistik“

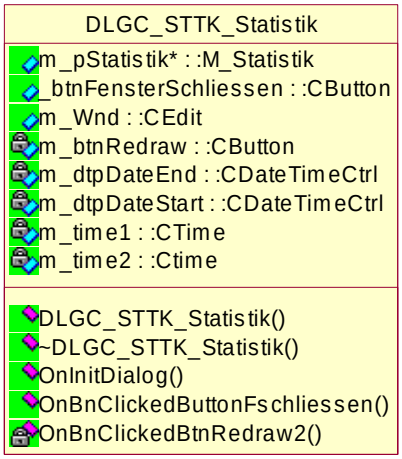


Abbildung 6.16 Klasse DLGC_STTK_Statistik

Klassename: DLGC_STTK_Statistik

Beschreibung: Diese Klasse stellt Methoden zur Grafikanzeige zur Verfügung..

Zugriff: Public

Variablen:

Öffentliche

`M_Statistik* pStatistik;`	Zeiger auf das Modul, woher die Daten stammen.
`CButton _btnFensterSchliessen;`	Schließt den Dialog .
`CEdit m_wnd;`	Platz auf dem Bildschirm zum Zeichnen.

Nicht Öffentliche

`CButton m_btnRedraw;`	Führt eine Aktualisierung der Grafik aus.
`CDateTimeCtrl m_dtpDateStart;`	Anfangsdatum, vom Benutzer definiert !
`CDateTimeCtrl m_dtpDateEnd;`	Enddatum, vom Benutzer definiert !
`CTime m_time1, m_time2;`	Datum aus der Controlen DateTimePicker. Zum Vergleich

Methoden:

Öffentliche

`DLGC_STTK_Statistik(M_Statistik* _pStatistik, CWnd* pParent = NULL);`	Standard Konstruktor
`virtual ~DLGC_STTK_Statistik();`	Standard Destruktor.
`virtual BOOL OnInitDialog();`	Initialisiert den Dialog .
`afx_msg void OnBnClickedBut-tonFschliessen();`	Fenster schließen.

Nicht Öffentliche

`afx_msg void OnBnClickedRedraw();`	Aktualisiert die Grafik, nur wenn das Anfangsdatum und das Enddatum angegeben sind

Oberklasse: CDiagDialog

Unterklassen: Keine

Beziehungen: Jedem statistischen Modul entspricht ein DLGC_Grafik Dialog.

Beziehung der Statistik Klasse mit der Target Klasse

Die Statistik Klasse wird direkt von der Diagnose abhängt und durch Instanzierung werden alle Informationen zur Anlage zur Verfügung stehen. Da jedes Target einem Statistik Modul enthalten wird, wird die Statistik Klasse für die gesamte Diagnose an die Basisklasse CDiagtarget gehängt.

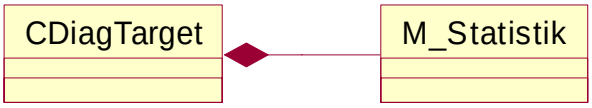


Abbildung 6.17 Aggregation zwischen Diagnose und Statistik

6.4 Datentypen und Konstanten

In dem Modul Statistik wurden folgenden Datentypen definiert:

Name	Beschreibung
`#define SECONDS_X_DAY 86400`	Entspricht der Anzahl von Sekunden pro Tag.
`typedef struct _register { CString m_name; CString m_meldung; int m_zeitpunkt; int m_f; int m_nr; }T_Register;`	Jede Fehlermeldung, die aus der Fehlerdatei ausgelesen wird, besteht aus einer Struktur mit folgenden Elementen: Name, Nummer, Zeitpunkt und entsprechender Zustand.
`typedef struct _errorlist { int m_nr; int m_status; CString m_text; }T_ErrorList;`	Aus der Datei „Meldung.csv“ werden die Meldungen identifiziert und in diese Struktur geladen.
`typedef struct _periode { CTime startDate; CTime endDate; } T_Period;`	Entspricht einem Zeitraum mit einem Anfangsdatum und einem Enddatum.
`typedef enum _rangeoftime{ kein = 0, tag = 86400, woche = 604800, monat = 2678400, quartal = 7948800, jahr =31536000, }T_PeriodTyp;`	Dieser Datentyp enthält die für das Statistik-Modul Einheit-Auswahl zur Untererteilung eines Zeitraums.
`typedef struct _infoPeriod{ CString unit; T_Period period; } T_InfoPeriod;`	Dieser Datentyp enthält die Information eines Zeitraumes, sowie Einheitsname und den entsprechenden Zeitraum.
`typedef struct _list{ CString label; int value; int nCol; }T_GraphTable;`	Struktur der Tabelle von Daten zur graphischen Darstellung.
`typedef enum _conexion { FTP = 1 // ... }anschlussTyp;`	Aufzählung von möglichen Verbindungen bei unterschiedlichen Bahntechniken.

6.5 GUI-Prototypen

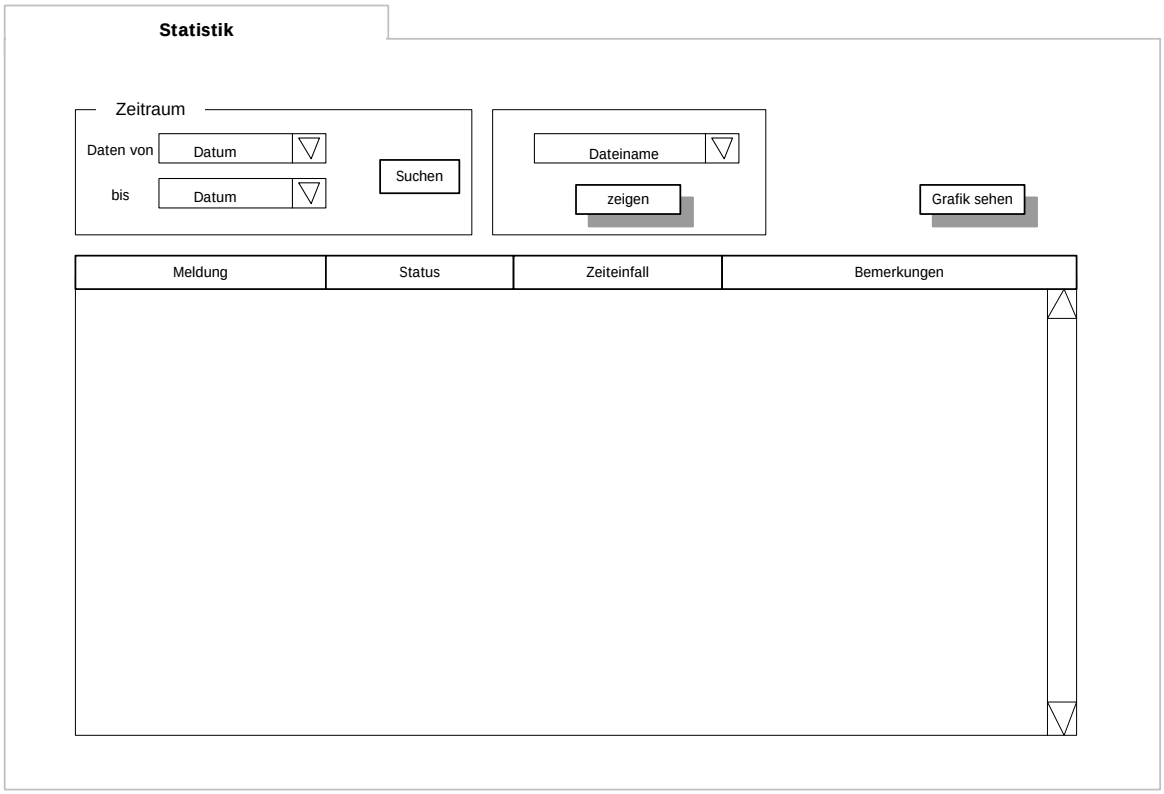


Abbildung 6.18 Dialog Übersicht

Dialogklasse: DLGC_STTK_Uebersicht

Beschreibung: Dieser Hauptdialog des statistischen Moduls gliedert sich als Karteikarte in die „Systemübergreifende Diagnose“ ein. Die zwei ersten DateTimeControls des Dialoges dienen zur Eingabe eines Beginn-Zeitraums und eines Ende-Zeitraums. Der Button „Suchen“ soll das eingetragene Datum validieren und die Funktionen zur Dateisuche aufrufen. Das Listfeld listet die Dateinamen der gefundenen Dateien und bietet die Wahl die Daten anzuzeigen. Mit dem Button „zeigen“ kann dann die ausgewählte Datei von dem DIMO geholt und angezeigt werden.

In dem Anzeigefenster werden die Daten bzw. Meldungen der ausgewählten Dateien angezeigt.

Der Button „Grafik sehen“ ruft den Dialog Statistik auf.

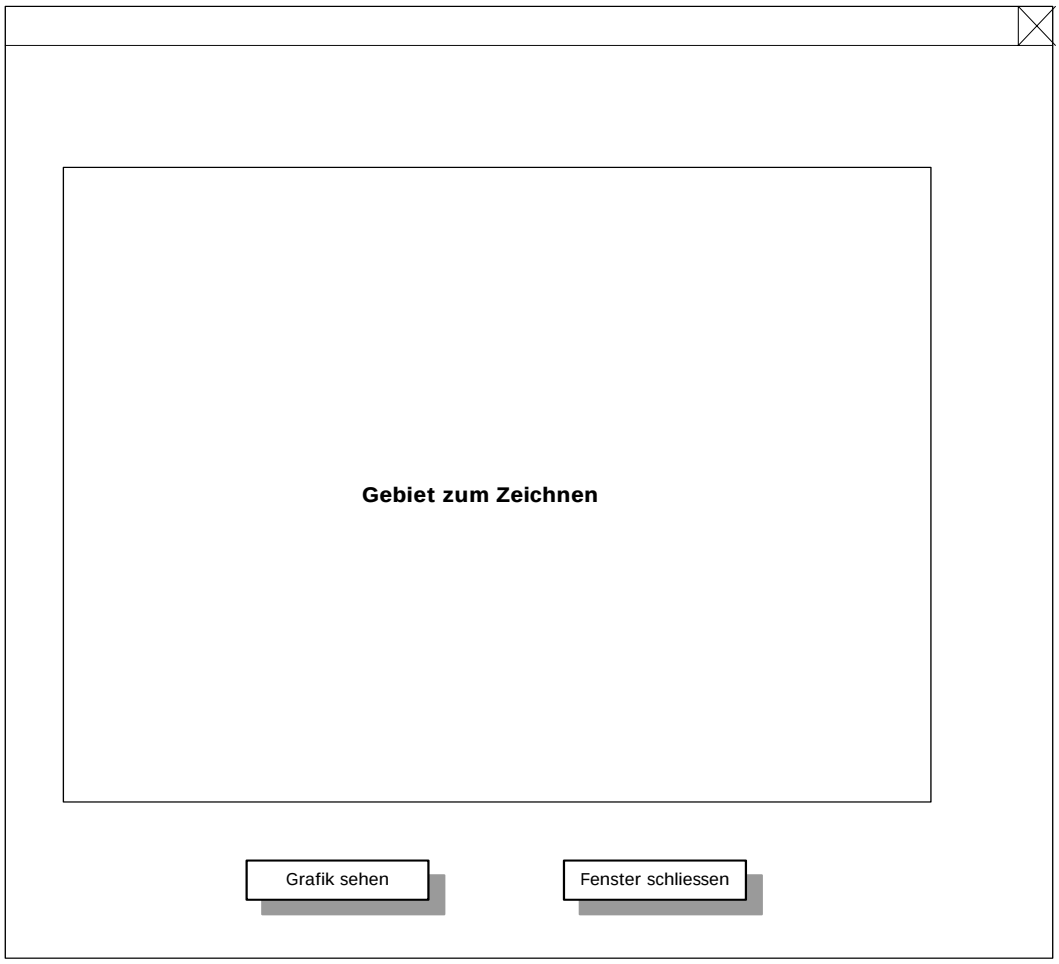


Abbildung 6.19 Dialog Statistik

Dialogklasse: DLGC_STTK_Grafik

Beschreibung: Der Dialog Statistik besteht aus einem Anzeigefenster, wo eine Grafik aus der Meldungen der angezeigten Fehlerdateien gezeichnet werden kann. Der Button „Grafik sehen“ ruft die Funktionen zum Zeichnen auf. Das Schließen des Dialoges erfolgt in zwei Formen, mit dem Button „Fenster schliessen“ oder mit dem oben stehenden Schließen-Feld.

7. Implementierung

Dieses Kapitel beschäftigt sich mit der Implementierung des in dieser Diplomarbeit ausgearbeiteten Moduls. In dieser Phase muss der Programmcode beschrieben werden, der das im Kapitel 6 entworfene Design für das statistische Modul in der systemübergreifenden Diagnose umsetzt.

In diesem Kapitel soll daher zunächst auf die Umgebung eingegangen werden, in der das Modul implementiert werden soll. Dazu zählen die im Kapitel 5 beschriebenen Nicht-Funktionalen Anforderungen, die in dem Modul eingebunden werden sollen. Das verwendete Betriebssystem, das zur Implementierung verwendete Tool, die Editoren und die festgelegten Programmierrichtlinien werden ebenfalls in diesem Kapitel beschrieben. Die Programmierrichtlinien erfolgen so, dass der Code auch für andere Entwickler leicht verständlich ist. Die einfache „Lesbarkeit“ des Codes ist während der Testphase und allen späteren Erweiterungen des Codes von großer Wichtigkeit.

Auf Grund der Einbettung des Moduls in die „systemübergreifende Diagnose“ war die Auswahl eines bestimmten Betriebsystems, sowie die Entwicklungsumgebung nicht notwendig, sondern bereits dadurch vorgegeben.

Umgebung und Tools

Betriebsystem

Wie die „Systemübergreifende Diagnose“ wurde auch das Statistik Modul auf dem Betriebsystem Microsoft Windows 2000 (Service Pack 4) implementiert.

#REQ-NFKT-01#REF#

Entwicklungsumgebung

Zur Implementierung wurde das Microsoft Visual Studio .NET 2003 mit der Programmiersprache Microsoft Visual C++ Version 7.1

#REQ-NFKT-04#REF#

Codeverwaltung

Die Quellcodeverwaltung und Dokumentation erfolgte über das Tool Rational Clear Case Explorer Version 2003

Programmierrichtlinien

In diesem Abschnitt sind die Programmierrichtlinien beschrieben, nach denen die Implementierung des Moduls erfolgte.

Allgemeine Regeln:

- Der Programmcode muss so verfasst werden, dass dieser intuitiv verstanden werden kann ohne unverhältnismäßig viele Kommentare lesen zu müssen oder den Autor des Codes zu fragen. Diese Regel gilt vor allem für Namen von Variablen und Methoden. Die Kommentare sind in der Definition jeder Klasse zu finden.

- Jede Datei muss an deren Beginn ein Statement mit dem Dateinamen, dem Autorennamen und einer kleinen Beschreibung in der folgender Form beinhalten:

```
//-----------------------------------------------------------------
// Datei:
// Autor:
// Beschreibung:
//-----------------------------------------------------------------
```

- Namen für neue Datentypen, Variablen, Membervariablen, Zeiger und Konstanten müssen so definiert werden, dass deren entsprechende Bezeichnungen erkennbar sind. In den folgenden Beispielen wird dies anschaulich dargestellt.

Neue Datentypen:	`Typedef _register{`
	`Cstring Name;`
	`int Value;`
	`}T_Register`

Variablen:	`LPTCSTR strFilename;`
Member variable:	`CString m_IPAdr;`
Zeiger:	`CInternetConnection* pFinder;`
Konstanten :	`#define SECONDS_X_DAY 8600`

Während der Implementierungsphase erfolgte die tatsächliche Umsetzung des entworfenen Moduls in einem ausführbaren Programmcode.
Die Prüfung der Funktionalität des Moduls ist in dem folgenden Kapitel beschrieben.
Eine Version der Systemweiten Diagnose mit dem umgesetzten und integrierten Statistik-Modul steht auf der beigefügten CD zur Verfügung.

8. Test und Validierung

Der Test des statistischen Moduls berücksichtigte einen Vergleich zwischen den in der Implementierungsphase realisierten Komponenten und dem im Kapitel 5 definierten Anforderungskatalog.

Validierung

#REQ-FKT-01#REF#
Es wurde ein Modul entwickelt, das die Basisfunktionalität zur Statistikbildung der Fehlermeldungen aus verschiedene Systemtechniken zur Verfügung stellt. Dafür dient die implementierte Klasse M_Statistik.

#REQ-FKT-02#REF#
Die Funktion zum Auslesen von Fehlerdateien wurde erstellt. In dem Fall der Bahnübergangtechnik SIMIS LC steht eine FTP-Verbindung zur Verfügung.

#REQ-FKT-03#REF#
Es wurden Funktionen implementiert, die die Gültigkeit und das Vorhandensein der Fehlerdateien in einem ausgewählten Zeitraum verifizieren.

#REQ-DAR-01#REF#
Durch graphische Oberflächen erfolgt eine tabellarische und eine graphische Darstellung der von der Statistik bearbeiteten Daten.

Black-Box Test

Es wurde ein „Black Box“ Test an dem Statistik Modul durchgeführt. Dazu wurde jedes Element von den Benutzeroberflächen aus dem Modul geprüft.
Die Ergebnisse des Testen sind wie folgt:

Dialog Uebersicht

Element	Ergebnisse
DateTimeControls	In Ordnung
DateTimeControls	In Ordnung
Der Button „Suchen“	In Ordnung
Listfeld	In Ordnung
Button „zeigen“	In Ordnung
Anzeigefenster	In Ordnung
Button „Grafik sehen“	In Ordnung

Dialog Statistik

Element	Ergebnisse
Anzeigefenster	Die Anzeige von mehr als 25 Daten soll bearbeitet werden.
Button „Grafik sehen“	In Ordnung
Button „Fenster schliessen“	In Ordnung

White-Box Test

Es wurde kein White-Box Test durchgeführt.

9. Zusammenfassung und Ausblick

Die Erstellung des Statistik-Moduls bietet die Möglichkeit, einen schnellen und einfachen Einblick von den in der Vergangenheit aufgezeichneten Fehlerdaten zu erhalten. Dazu erleichtert die dynamische Verwaltung von Zeiträumen, die Präzision in Darstellung frei zu wählen.
Die Vorteile dieses Moduls liegen darin, Schwachstellen frühzeitig zu identifizieren und sie möglichst im Vorfeld zu vermeiden.

Einige Erweiterungsmöglichkeit findet sich in:

- der Implementierung der speziellen Statistik Module für anderen Techniken der „Systemweite Diagnose“.
- die Implementierung von Funktionen zur unterschiedlichen Auswertungen und Anzeige von Fehlerinformationen, so wie die Möglichkeit Fehler zu prognostizieren.
- die Projektierbarkeit der einzelne Fehlermeldungen in Statistikbildung.
- Einheitliche Statistik Modul für alle Techniken.

.

Literaturverzeichnis

1. Jürgen Ott, H.: *Software-Systementwicklung, Praxisorientierte Verfahren und Methoden*, Carl Hanser Verlag München Wien, 1991.
2. Pomberger, G.: Blaschek, G.: *Software Engineering, Prototyping und objektorientierte Software-Entwicklung*. 2 Auflage. Carl Hanser Verlag München Wien, 1996.
3. Oestereich, B.: Objekt-orientierte Software-entwicklung, *Anayse und Design mit der Unified Modeling Language.* 5., völlig überarbeitete Auflage. Oldenbourg Verlag München Wien, 2001
4. Prosise, J.: Programming Windows with MFC. Second Edition. Microsoft Press Books. USA ,1999
5. L. Sachs, Statistische Methoden ISBN 3540520252
6. Sudhaus gen. Rohr. PEACC Band 1. Grundlagen zum PEACC. Revision 4. 18.01.2005.
7. Siemens AG Transportation System: *Das Bahnübergangssystem SIMIS LC,*
8. Siemens AG Transportation System: *Diagnose Modul,* DIMO_Intro.pps, 2003
9. Siemens AG Transportation System: *Bahnübergangs-Sicherungssystem SIMIS LC;* http://references.transportation.siemens.com/refdb/showReference.do?r=353
10. Siemens AG Transportation System: Grundlagen zum *PEACC* http://automation.ts.siemens.de/cfm/peacc-hw/downloads/peacc1.pdf,
11. http://msdn.microsoft.com
12. http://automation.ts.siemens.de/prozesse_qm/prozesse/peacc_r3/_home/index.htm
13. http://www.ivev.bau.tu-bs.de/
14. http://begriffsportal.de/Eisenbahnkreuzung
15. http://intra.ts.siemens.de
16. http://www.stellwerke.de/signal/deutsch/bue.html
17. http://references.transportation.siemens.com/

18. http://www.hochschulstellenmarkt.de/info/h/ha/ha_ufigkeitsverteilung.html
19. http://www.statistik.tuwien.ac.at/public/dutt/vorles/inf_bak/node6.html
20. http://www.lernstats.de

Anhang 1: PEACC-Modell

Die Abkürzung PEACC steht für **P**rodukt-**E**ntwicklung **a**uf der Basis von **C**ENELEC und **C**MM. CENELEC steht darin für eine bestimmte, die Entwicklung von Bahntechnik betreffende Gruppe von CENELEC-Normen; CMM (Capability Maturity Model) ist ein aus den USA stammendes, inzwischen international gebräuchliches Modell zur stufenweise Bewertung und Verbesserung von Entwicklungsprozessen. Das Modell wurde eigens von Siemens VT entwickelt und dient eine klare Orientierung in der Projektentwicklungsprozess (Siehe quelle 16).

Phasenmodell

Dieses Model ist zwar V-förmig dargestellt, da sich so die Rückflüsse der späteren zu den früheren Phasen am besten darstellen lassen.

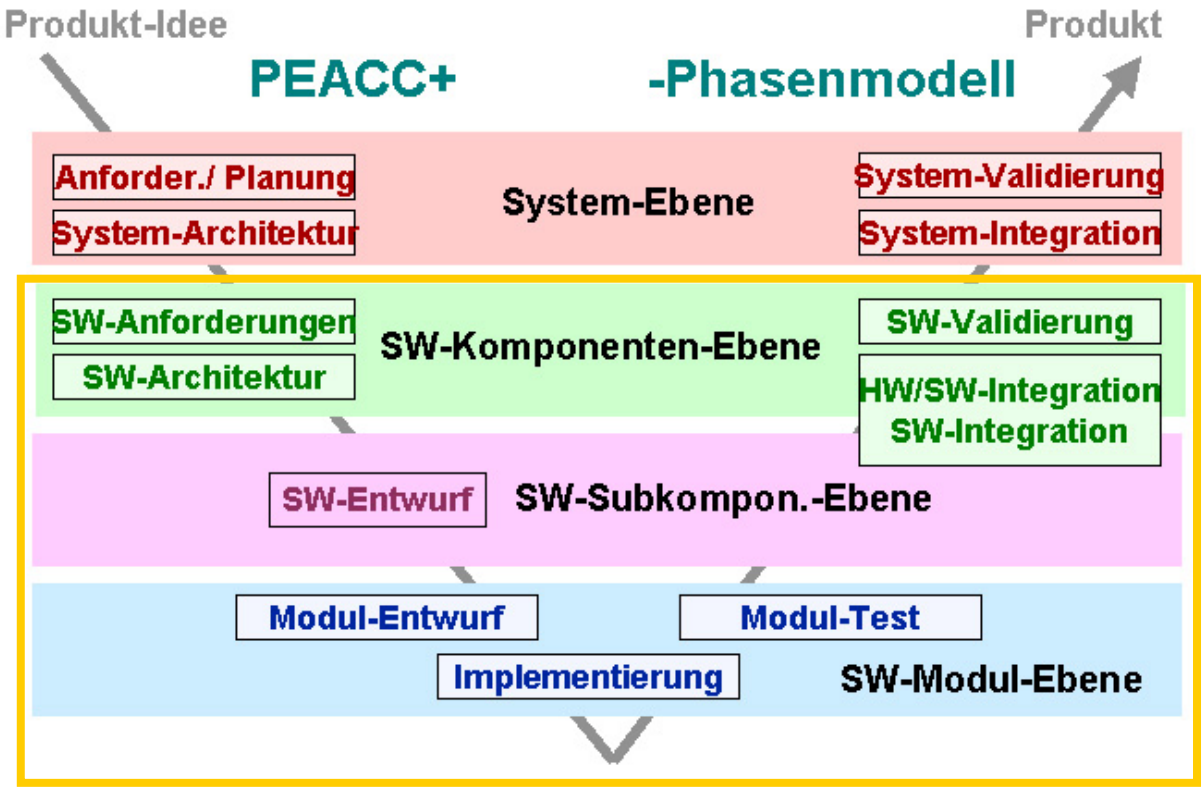


Abbildung A.1 PEACC Modell(Bild aus Schulungsmaterial genommen)

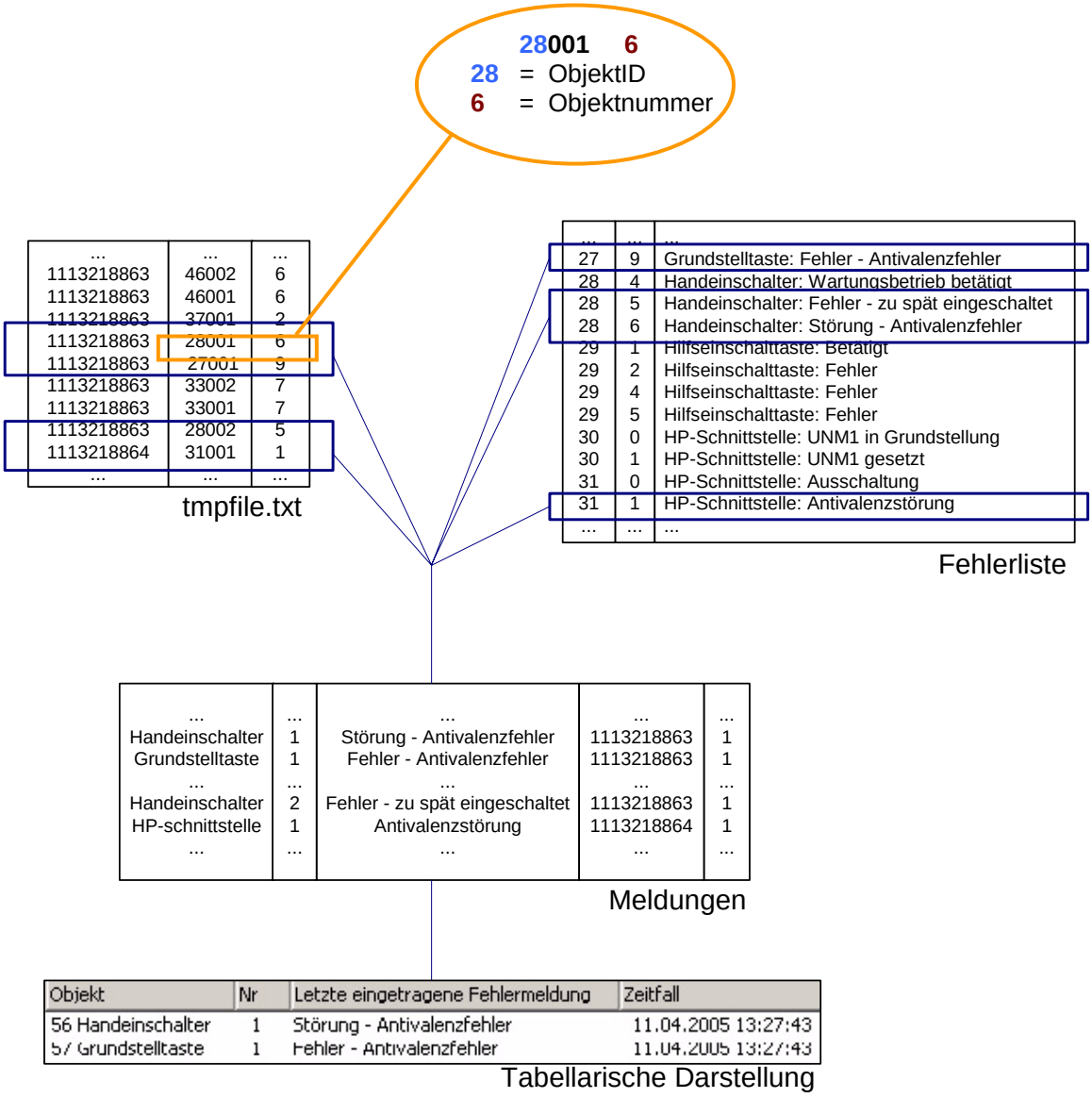
Für die Realisierung dieser Diplomarbeit wurde die Ebene SW-Komponenten-Entwicklung berücksichtig(S. Abb. A.1), dessen entsprechen folgende Phasen:

- **Anforderungsphase:** In dieser Phase besteht aus der Spezifikation aller Anforderungen (**was** soll realisiert werden?), Planung der Testfallen, mit denen die Erfüllung nachgewiesen werden kann;
- **Architekturphase:** besteht aus der Umsetzung der Anforderungen in eine Produktarchitektur (**wie** soll realisiert werden ?) mit Zuordnung der einzelnen Anforderungen aus der Anforderungsspezifikation zu den Architekturelemente (Grobenentwurf);
- **Design (Entwurfs-)phase:** Hier wird die Verfeinerung des Architekturelements in eine Modulstruktur (Feinentwurf); Zuordnen der Anforderungen auf die einzelnen Module;
- **Implementierungsphase:** Umsetzung der Designspezifikation in ein funktionierendes, testbares HW- bzw. SW- Modul;
- **Testphase:** Test des realisierten Moduls basierend auf dem Testplan aus der Designphase (Funktionstest);
- **Integrationsphase:** Zusammenfügen aller in der Architektur- und Designphase geplanten Architekturelemente und Überprüfen der Gesamtfunktionalität und der Wechselwirkung zwischen den Architekturelementen;
- **Validierierungsphase:** Überprüfen mit Hilfe der in der Anforderungsphase spezifizierten Testfälle, ob alle spezifizierten Anforderungen erfüllt sind.

Anhang 2: Beispiel

Tabellarischen Darstellung

Folgendes Beispiel stellt den Ablauf der Informationsbearbeitung dar. Das Datum 11.04.2005 ist angegeben. Danach realisiert das Modul eine Suche nach Dateien, die unter diesem Datum gespeichert wurden. In diesem Beispiel ist die Suche erfolgreich gewesen und wurde es die Datei „tmpfile.txt“ mit den gefundenen Dateien erstellt. Ebenfalls ist die Fehlerliste mit den Informationen aus der Datei „“Meldung.csv“ erstellt. Mittels eines Vergleiches von der ID und den Zustandinformation jeder Meldung mit der Fehlerliste, werden die Zellen der Tabelle mit den Meldungen ausgefüllt.



Die Meldungen können dann vom Benutzer angesehen werden. Die Form, in dem diese Darstellung erfolgt, ist in der folgenden Abbildung A.1 gezeigt:

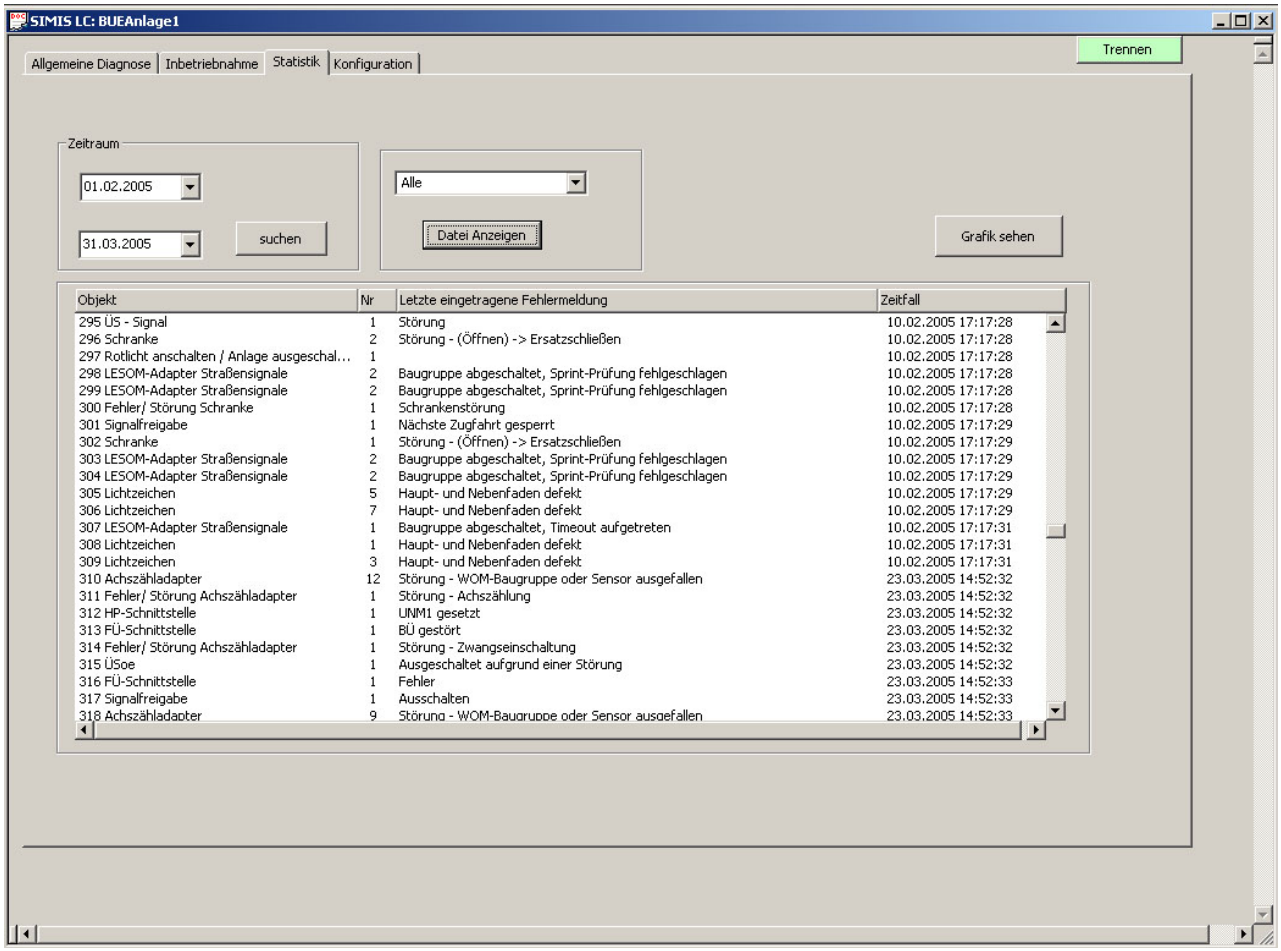


Abbildung A.2 Anzeige von Fehlermeldungen auf der „systemweite Diagnose“.

Graphische Darstellung

Für eine Grafische Darstellung wird die Häufigkeit jedes einzeln Objekt gerechnet und in einer Grafik dargestellt. Die folgende Abbildung zeigt ein Beispiel dafür.

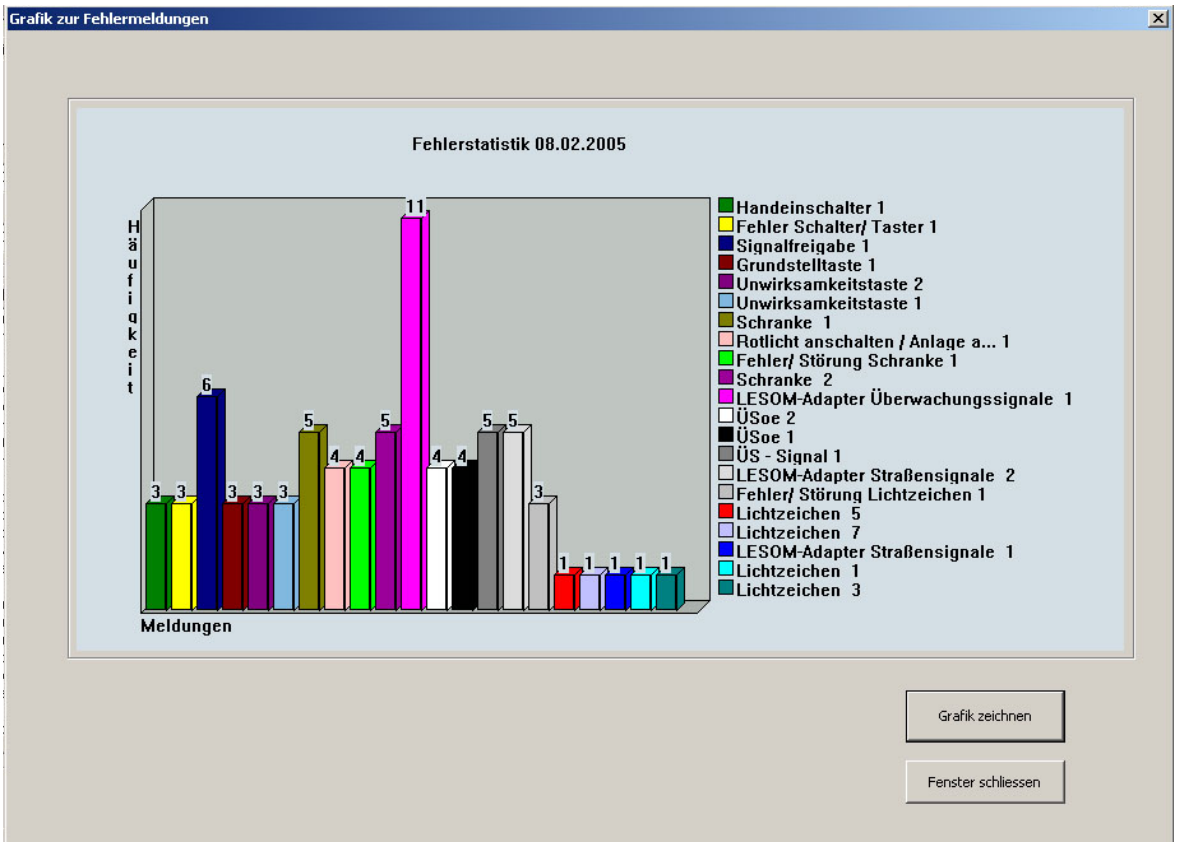


Abbildung A.3 Graphische Anzeige von Fehlermeldungen auf der „systemweite Diagnose“