FAKULTÄT FÜR !NFORMATIK

Feldfehlerprognose

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Informatik

ausgeführt von

Alexander N. Menches
Matrikelnummer 8525026

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: A. o. Univ. Prof. Dipl.-Ing. Dr. techn. Gerald Futschek

Wien, 16.10.2008 _____ _____

(Unterschrift Verfasser) (Unterschrift Betreuer)

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Wien, 16. Oktober 2008 _____

Danksagung

Ich danke meinem Betreuer a. o. Univ. Prof. Dipl.-Ing. Dr. techn. Gerald Futschek für die fachlichen Anregungen und wohlwollende Beurteilung dieser Diplomarbeit. Er war es auch, der durch seine einschlägige Lehrveranstaltung mein Interesse an diesem Thema geweckt hat.

Ich bedanke mich bei meinem Kollegen Dr. Alfons Bridi für oftmaliges gutes Zureden und seine tatkräftige Unterstützung, schon lange bevor diese Arbeit Gestalt angenommen hat.

Weiters danke ich meinem ältesten Freund und Wegbegleiter a. o. Univ. Prof. Dr. med. Christian Nasel, der mir eine mustergültige akademische Karriere vorgelebt hat und niemals müde wurde, mir vor Augen zu halten, dass ein abgeschlossenes Studium als Job-Versicherung gute Dienste leisten kann.

Ich danke meinem Kollegen und Freund Joachim Grüneis, der mein akademisches Schicksal teilt und daher auch immer Ansprechpartner und Ansporn zugleich war.

Ich bedanke mich auch bei meinen Eltern, die immer an mich geglaubt haben und durch ihre großzügige Unterstützung einen wesentlichen Anteil an meiner akademischen Laufbahn haben.

Last but not least bedanke ich mich bei meiner geliebten Frau Michaela, die mich durch Zeiten schwerer Krankheit ebenso begleitet hat wie durch die Höhen und Tiefen meiner beruflichen und akademischen Karriere. Mit ihren exzellenten Sprachkenntnissen hat sie auch zur besseren Lesbarkeit dieser Arbeit beigetragen.

Für Fabio

Kurzfassung

Diese Arbeit beschäftigt sich mit Möglichkeiten der Prognose von Fehleranzahlen, Fehlerdichten bzw. Fehlerraten in Software-Systemen. Dazu werden drei Hypothesen zur Diskussion gestellt. Anhand des Begriffes des Feldfehlers werden zunächst relevante Studien vorgestellt, analysiert und den Hypothesen gegenübergestellt. Unter einem Feldfehler versteht man dabei einen Software-Fehler, der auch nach der Veröffentlichung eines Software-Systems noch in diesem enthalten ist.

Anschließend wird im Rahmen einer Fallstudie aus einem realen Projektumfeld eine Feldfehlerprognose durchgerechnet und analysiert. Auch die Fallstudie wird den Hypothesen dieser Arbeit gegenübergestellt.

Wesentliche Voraussetzung für verlässliche Feldfehlerprognosen ist die ausreichende Verfügbarkeit einer breiten Palette von Metriken sowie der zugehörigen Fehlerdaten auch aus der Vergangenheit. Wird dies nicht bereits zu Beginn eines Projektes oder einer Software-Produktentwicklung berücksichtigt, kommt es durch eingeschränkte Verfügbarkeit von Metriken (kann man auch als Prediktoren des Prognosemodells auffassen) bzw. einer zu geringen Zahl von historischen Daten zu schlechten Feldfehlerprognosen.

Abstract

This diploma thesis deals with the possible prognosis of field defects, field defect density resp. field defect rates within software systems. For this purpose three hypotheses are put up for discussion. Relevant studies on the basis of the term “field defect” are being introduced, analysed and compared with the hypotheses. A field defect means a software error which is still immanent within a software system after its release.

Subsequently a field defect prognosis is being calculated and analysed by means of a case study coming from a real software project environment. The case study is also being compared to the hypetheses.

The most essential preconditions for reliable field defect prognoses are the availability of a sufficient amount of metrics as well as the corresponding error data from the past. If these requirements are not met right from the beginning of a project or software development the limited availability of metrics (which can also be understood as predictors of the prognosis model) resp. the too small amount of historical data will lead to inferiore field defect prognoses.

Inhaltsverzeichnis

Abkürzungsverzeichnis

LOC	Lines of Code (Anzahl der Code-Zeilen)
KLOC	1000 Lines of Code
ARE	Absolute-Relative-Error
BIC	Bayessches Informationskriterium
CORE	Customer Oriented Reliability Estimate
TDD	Test Driven Development
STREW	Software Testing and Reliability Early Warning
PLA	Principal Component Analysis (Diskriminanzanalyse)
J48	Algorithmus aus dem Bereich des Entscheidungsbaum-basierten Lernens
MDP	(NASA) Metrics Data Program
KDV	Kommerzielle Datenverarbeitung
ANOVA	Analysis of Variance

Abbildungsverzeichnis

Formelverzeichnis

Tabellenverzeichnis

Kapitel 1

1 Einleitung

1.1 Überblick

Die Menschheit wollte schon immer in die Zukunft blicken, um ihr Schicksal zu beeinflussen. Eine recht amüsante Darstellung dazu bieten Uderzo und Goscinny in Asterix Band 19 „Der Seher“ (sieheAbbildung 1-1).

Im 21. Jahrhundert stellt die Herstellung von Computer-Software einen wesentlichen Wirtschaftszweig dar. Viele Bereiche unseres Lebens sind dabei durchaus von der Funktionsfähigkeit sogenannter „Mission-Critical“ Computersysteme (Energieversorgung, Verkehr, Kommunikation, etc.) abhängig geworden.

Die Herstellung von Computer-Systemen ist eine Disziplin des Ingenieurswesens und wird auch als Software-Engineering bezeichnet. Leider sind viele Ergebnisse von Teilschritten im Rahmen des Software-Engineering-Prozesses nicht unmittelbar „begreifbar“, wie das z.B. die Teilergebnisse eines Tischlers bei der Herstellung eines Möbelstücks sind. Software-Teilergebnisse sind oft nur in Prosa verfügbar, mit allen Möglichkeiten des Missverständnisses zwischen Auftraggeber und Ersteller. Die Herstellung von Software-Systemen ist durch die hohen Ansprüche an die Kommunikation zwischen allen Beteiligten daher wesentlich fehleranfälliger als ein herkömmlicher „begreifbarer“ Herstellungsprozesse. Ab einem gewissen Komplexitätsgrad ist es nach wie vor nur mit hohem Aufwand bzw. gar nicht möglich, fehlerfreie Software-Systeme herzustellen.

Um nun wenigstens einen möglichst wahrheitsgetreuen Eindruck von der Qualität eines Software-Systems zu erhalten, wäre es schön zu wissen, wie viele Fehler denn in einem fertigen Software-System enthalten sind.

Abbildung 1-1: Aus Asterix Band 19 "Der Seher"

Seit vielen Jahren beschäftigen sich Wissenschafter daher auf internationaler Ebene mit der Fehlerprognose von Software-Systemen. Grundsätzlich kann man hier zwischen Methoden zur Fehlerprognose in Source-Codes nach verteilten Code-

Inspektionen[1] und Metrik-basierten Fehlerprognosen in Software-Systemen unterscheiden[2].

Für die Metrik-basierte Fehlerprognose in fertigen Software-Systemen kommen dabei sowohl statistische Methoden, wie z.B. verschiedene Regressionsmodelle, als auch Methoden des maschinellen Lernens, wie z.B. Neuronale Netzwerke und Entscheidungsbäume zum Einsatz. In der im Rahmen dieser Arbeit untersuchten Literatur besteht aber noch immer keine eindeutige Meinung, welche die besten Modelle zur Fehlerprognose sind. Möglicherweise gibt es nicht das eine Modell für alle Anwendungsfälle, vielmehr dürfte für jeden Einzelfall ein geeignetes Modell zu finden und zu verwenden sein. Auch die Aussagen darüber, was nun eine „gute" Prognose ist, lassen keine allgemein gültige Definition der Prognosegenauigkeit zu. Gemeinsam mit der Auswahl geeigneter Prediktoren müssen sowohl das Modell, als auch die notwendigen bzw. verlangten Genauigkeiten projektspezifisch festgelegt werden.

Die vorliegende Arbeit beschäftigt sich mit der Metrik-basierten Fehlerprognose in Software-Systemen, im Weiteren auch Feldfehlerprognose genannt. Zunächst werden die Grundbegriffe der Feldfehlerprognose definiert – z.B. was ist ein Feldfehler? Unterschied zwischen Feldfehleranzahl und Feldfehlerdichte, etc. - und der Nutzen der Feldfehlerprognose umfassend dargelegt. Danach folgt ein Streifzug durch die in der Literatur beschriebenen Ansätze zur Fehlerprognose. Im praktischen Teil wird eine Fehlerprognose eines großen Software-Systems anhand multilinearer Regressionsmodelle schrittweise berechnet, beschrieben und anhand der im Kapitel „Ziel" aufgestellten Hypothesen diskutiert. Zum Abschluss gibt es noch praktische Hinweise zur Anwendung von Fehlerprognosen in realen Software-Projekten anhand konkreter Verbesserungsvorschläge zur Fallstudie.

1.2 Ziel der Arbeit

Die Arbeit soll dem interessierten Leser Grundlagen zu den nötigen Rahmenbedingungen und der Durchführung von Feldfehlerprognosen vermitteln, auf deren Basis eine weiterführende Auseinandersetzung mit dem Thema möglich ist.

---

[1] z.B. [Biff00], [Biff01], [Biff00a], [Biff02], [Biff03], [Rune98], [Padb02], [Ragg02], [Thel04], [Chan05]

[2] z.B. [Sany92], [Moha02], [Bai03], [Refo03], [Li04], [Boet05], [Chal05], [Koru05], [Naga05], [Sher05], [Li06], [Knab06], [Wang06]

Da sich die Arbeit auch mit den praktischen Aspekten des Einsatzes von Feldfehlerprognosen in einem realen Projektumfeld beschäftigt, soll sie einen Anreiz bieten, Feldfehlerprognosen als integralen Bestandteil von Software-Engineering-Projekten und als interessante Methode zur Verbesserung der Software-Qualität zu sehen.

Im Rahmen dieser Arbeit werden dazu folgende Hypothesen zur Feldfehlerprognose zur Diskussion gestellt:

1. Die Qualität von Fehlerprognosen hängt von der Wahl der geeigneten Prediktoren ab.
2. Nachträgliche Datengewinnung führt durch eingeschränkte Verfügbarkeit von geeigneten Prediktoren zu Feldfehlerprognosen schlechter Qualität.
3. Die Rahmenbedingungen für die Feldfehlerprognose müssen bereits zu Beginn der Software-Entwicklung festgelegt werden, um Feldfehlerprognosen hoher Qualität zu erhalten.

Kapitel 2

2 Definition und Nutzen der Feldfehlerprognose

2.1 Definition Feldfehler

Unter einem Feldfehler versteht man einen Software-Fehler, der auch noch nach Veröffentlichung eines Software-Systems in diesem enthalten ist. Ein Feldfehler ist daher ein Fehler, der während des Entwicklungs- und Testprozesses nicht gefunden und behoben wurde. Fehler, die vor Veröffentlichung eines Software-Systems in diesem enthalten sind, werden in dieser Arbeit einfach als Fehler bezeichnet. Unter einem Software-Fehler versteht man dabei im Allgemeinen die Abweichung der Funktionsweise eines Software-Systems von seiner Spezifikation.

Der Begriff Software-System wird in dieser Arbeit zur Bezeichnung einer für einen bestimmten Anwendungszweck geschaffenen Menge an Computerprogrammen verwendet. Subsysteme oder auch einzelne Programme eines Software-Systems werden als Software-Komponenten bezeichnet.

Unter einer Software-Release wird in dieser Arbeit die Veröffentlichung einer bestimmten Version eines Software-Systems verstanden.

2.2 Fehlermetriken

Folgende Fehlermetriken werden in dieser Arbeit behandelt:

- Anzahl Feldfehler/Fehler: Anzahl der in einem Software-System oder in einer Software-Komponente enthaltenen Fehler.
- Feldfehlerdichte/Fehlerdichte: Anzahl der in einem Software-System oder in einer Software-Komponente enthaltenen Fehler dividiert durch eine verfügbare Software-Metrik (vorzugsweise Größenmetriken).
- Feldfehlerrate/Fehlerrate: Anzahl der in einem Software-System oder in einer Software-Komponente auftretenden/gefundenen Fehler je Zeiteinheit.

Im Prinzip handelt es sich hier um den zeitlichen Verlauf des Auftretens von Feldfehlern.

Sind historische Daten verfügbar, so kann man für die Fehlermetriken Anzahl Feldfehler und Feldfehlerdichte auch Trend- oder Deltawerte bilden, indem man die Differenz einer Metrik jeweils zwischen zwei zeitlich aufeinanderfolgenden Software-Releases bildet.

2.3 Nutzen

Der ureigenste Sinn der Herstellung eines Software-Systems (Software-Entwicklungs- oder Software-Engineering-Prozess) ist es, das Software-System möglichst genau den Wünschen der Anwender anzupassen. Je besser ein Software-System den Wünschen der Anwender entspricht, desto höher ist seine Qualität. Im Detail setzt sich die Qualität eines Software-Systems natürlich aus mehreren Faktoren zusammen (z.B. einfache Benutzbarkeit, Reaktionszeit, Laufzeit, etc.). Ein wesentlicher Faktor ist aber die Verlässlichkeit des Systems und diese hängt wieder unmittelbar mit der Anzahl der in dem Software-System enthaltenen Feldfehler zusammen.

Die Anzahl der in einem Software-System enthaltenen Feldfehler ist durch folgende Faktoren bestimmt:

- Anzahl und Qualität der verfügbaren Software-Ingenieure
- Qualität der Kommunikation zwischen Software-Ingenieuren und Anwendern (in diesem Punkt geht es vor allem um das Fachwissen, das die Software-Ingenieure im Anwendungsgebiet haben, sowie um das Verständnis um informationstechnische Möglichkeiten auf Seiten der Anwender)
- Software-Engineering-Prozess (Prozessmodell, Qualitätssicherungsmaßnahmen)
- Verwendete CASE-Werkzeuge
- Verwendete Entwicklungsumgebung
- Verwendete Systemarchitektur

Einige dieser Faktoren können auch während des Entwicklungsprozesses beeinflusst werden, wie z.B. Anzahl und Qualität der Software-Ingenieure, Entwicklungsprozessmodell sowie Art und Anzahl der Qualitätssicherungsmaßnahmen. Ob, wann und wie diese Beeinflussung vorgenommen wird, ist in der Regel die Entscheidung des Projektmanagers. Der

Projektmanager muss dabei immer auch das Gleichgewicht von Qualität, Zeit und Budget (in Anlehnung an den sogenannten „Triple Constraint“ aus Scope, Zeit und Budget im Projektmanagement) herstellen. Zeit und Budget kann man durch geeignete Projekt-Kostenrechnung kontrollieren. Aber wie kann man die aktuelle bzw. künftige Qualität eines in Entwicklung befindlichen Software-Systems einschätzen?

Da Software-Systeme in der Praxis unter normalen Umständen nicht fehlerfrei hergestellt werden können, ist jedenfalls auch nach Veröffentlichung eines Software-Systems mit einem gewissen Wartungsaufwand zu rechnen. Dieser Wartungsaufwand ist von der Qualität des Software-Systems und damit in hohem Maße von den in diesem Software-System noch enthaltenen Feldfehlern abhängig. Wie kann ein Projektmanager den zu erwartenden Wartungsaufwand und damit die notwendigen Ressourcen abschätzen und planen?

Eine Lösung für beide Fragen ist der Einsatz von geeigneten Methoden zur Fehler- bzw. Feldfehlerprognose. Da man auch den Phasenübergang innerhalb eines Software-Entwicklungsprozesses als Veröffentlichungsprozess auffassen kann, können die Methoden zur Feldfehlerprognose auch zur Kontrolle der Software-Qualität während der Entwicklung verwendet werden. Selbstverständlich kann man mit diesen Methoden auch eine Aussage über die Anzahl der noch enthaltenen Feldfehler nach Veröffentlichung des Software-Systems machen.

Im ersten Fall verhelfen Methoden zur Feldfehlerprognose daher dem Projektmanager zu fundierten Entscheidungen bzgl. der Einhaltung des Gleichgewichts zwischen Qualität, Zeit und Budget. Im zweiten Fall kann der Projektmanager den zu erwartenden Wartungsaufwand fundiert einschätzen und damit die notwendigen Wartungsressourcen optimal planen.

Um in den Software-Engineering-Alltag einziehen zu können, müssen die Methoden zur Feldfehlerprognose allerdings einfach einsetzbar sein, sowie verwertbare Ergebnisse liefern. Die Frage nach solchen Ergebnissen sowie den Bedingungen, unter denen diese erzielt werden können, wird in dieser Arbeit in den Kapiteln 3 und 4 noch ausführlicher behandelt.

Kapitel 3

3 Grundlagen der Feldfehlerprognose

3.1 Allgemeines

Gegliedert nach den im Kapitel 2.2 erwähnten Fehlermetriken, werden in den folgenden Kapiteln jeweils die zugehörigen Grundlagen anhand in der Literatur verfügbarer Studien vermittelt.

Zunächst beschäftigt sich Kapitel 3.2 mit der Vorhersage der Feldfehleranzahl. Dieses Kapitel bildet eine wesentliche Grundlage für den im Kapitel Fallstudie beschriebenen praktischen Teil dieser Diplomarbeit.

Die Kapitel 3.3, 3.4 und 3.5 beschäftigen sich mit der Vorhersage von Feldfehlerdichten. Die Kapitel 3.3 und 3.4 decken dabei auch den Einsatz von auf maschinellem Lernen basierenden Algorithmen zur Feldfehlerprognose ab.

Die Kapitel 3.6 sowie 3.7 beschäftigen sich schließlich mit der Prognose von Feldfehlerraten.

3.2 Prognose der Feldfehleranzahl

Paul Luo Li, James Herbsleb und Mary Shaw vom Institute for Software Research, International Carnegie Mellon University, Pittsburgh. PA sowie Brian Robinson von ABB, Inc., Wickliffe, OH berichten in [Li06] über Erfahrungen und Ergebnisse bei der Einführung von Feldfehlerprognose zur Priorisierung von Software-Komponententests bei ABB.

[Li06] wollte in Zusammenhang mit der Einführung von Feldfehlerprognose folgende Fragen beantworten:

- Welche Methoden der Feldfehlerprognose eignen sich, um eine Risikoanalyse hinsichtlich der Veröffentlichung einer bestimmten Software-Produktversion sowie eine Priorisierung der Software-Komponententests durchführen zu können?

- Wie kann die Genauigkeit von Feldfehlerprognosen unter Berücksichtigung der Veränderung von Umgebungsparametern über mehrere Versionen hinweg überprüft werden?
- Welche Informationen sind notwendig, um qualitativ brauchbare Feldfehlerprognosen zu erhalten bzw. ist eine vernünftige Feldfehlerprognose auch mit unvollständigen Informationen möglich? (Diese Fragestellung deckt sich auch mit den Zielsetzungen dieser Diplomarbeit.)
- Welche Modellierungsmethoden sind für eine gute Feldfehlerprognose am besten geeignet? Im Speziellen wurde die Frage untersucht, welche Modellierungsmethode für die Prognose der Anzahl Feldfehler im ersten Jahr nach Inbetriebnahme am besten geeignet ist.
- Welche Prediktoren sind für eine gute Feldfehlerprognose nötig?

[Li06] untersuchte zwei Echtzeitsysteme (ein Monitoring System – Produkt A sowie ein Controller Management System- Produkt B) bei ABB. Von Produkt A standen 13 Versionen über einen Zeitraum von 5 Jahren zur Verfügung, von Produkt B waren es 15 Versionen über einen Zeitraum von 9 Jahren.

Von Produkt A war bekannt, dass es einen Umfang von etwa 300 KLOC gehabt hat. Von etwa 40 verschiedenen Personen sind bis zum Zeitpunkt der Untersuchung 127.000 Änderungen durchgeführt worden. Von Produkt B war bekannt, dass es einen Umfang von etwa 780 KLOC gehabt hat. Für dieses Produkt standen keine Zahlen bzgl. Anzahl Änderungen und Anzahl Personen zur Verfügung. Expertenschätzungen zufolge sind etwa 50 Personen mit der Pflege und Wartung des Produktes beschäftigt gewesen.

Die für die Feldfehlerprognose notwendigen Basisdaten stammten aus dem Serena-Tracker (Befunddatenbank) – das Produkt heißt mittlerweile Merant-Tracker und wird auch im Rahmen der in dieser Arbeit enthaltenen Fallstudie (siehe Kapitel Fallstudie) verwendet –, aus Microsoft Visual SourceSafe sowie aus der Befragung verschiedener Experten bei ABB.

Ein Feldfehler wurde gem. [Li06] bei ABB wie folgt definiert: Ein Feldfehler ist ein Befund in der Befunddatenbank, der als Fehler eingestuft wurde und zum Zeitpunkt der Veröffentlichung einer Produktversion noch in dieser enthalten ist.

Um eine geeignete Modellierungsmethode auswählen zu können, muss man sich erst fragen, welche Modellierungsmethoden grundsätzlich zur Verfügung stehen. [Li06] ging von folgenden 4 Arten der Modellierung aus:

- Beziehungen: Es werden Beziehungen zwischen Prediktoren und Feldfehlern hergestellt, z.B. eine Beziehung zwischen dem CMM-Level einer Organisation und der Anzahl von Feldfehlern in Projekten dieser Organisation[3].
- Klassifizierung: Es wird eine Wahrscheinlichkeit angegeben, ob die Anzahl der Feldfehler über einem bestimmten Grenzwert liegt oder nicht. Damit können Software-Module als riskant (wird mindestens einen Feldfehler enthalten) oder nicht riskant (enthält keinen Feldfehler) eingestuft werden[4].
- Mengen: Es wird versucht, die Anzahl der Feldfehler direkt zu prognostizieren (z.B. durch Verwendung neuronaler Netzwerke).
- Fehlerraten: Vorhersage einer Feldfehlerrate über einen bestimmten Zeitraum für ein bestimmtes Software-Produkt.

[Li06] stellte fest, dass viele Studien besonderen Wert auf die Untersuchung der Genauigkeit eines Modells zur Fehlerprognose legen. [Li06] legte aber größeren Wert auf die Aussagekraft von Prognosemodellen, vor allem hinsichtlich der Auswahl von geeigneten Prediktoren.

Welche Arten von Prediktoren stehen nun nach [Li06] zur Verfügung:

Produktmetriken: Messen die Eigenschaften eines Zwischen- oder Endproduktes eines Software-Entwicklungsprozesses (siehe Tabelle 3-1).

Tabelle 3-1: Produktmetriken [Li06]

Grundlage	**Prediktor**
Komplexität	Anzahl der möglichen Programmpfade Anzahl Knoten (überlappende Sprungbefehle) Maximale Schachtelungstiefe Zyklomatische Komplexität Halstead's Vocabulary

[3] Vergleiche auch [Hart00]

[4] Vergleiche auch [Kosh01]

Umfang	Anzahl Klassen Anzahl Dateien Anzahl Funktionen Lines of Code (LOC) Anzahl Operationen Anzahl Operanden Halstead's Length Halstead's Volume
Verschiedene Statements	Anzahl verschiedener Operationen
Mentaler Aufwand zur Herstellung eines Programmes	Halstead's Difficulty Halstead's Effort
Modularität bzw. Tiefe des Syntax-Baumes	Anzahl Basisklassen Abhängigkeit der einzelnen Klassen untereinander Tiefe des Vererbungsbaumes

Entwicklungsmetriken: Messen die Eigenschaften des Entwicklungsprozesses (siehe Tabelle 3-2)[5].

Tabelle 3-2: Entwicklungsmetriken [Li06]

Grundlage	**Prediktor**
Anzahl von Fehlern, die vor Veröffentlichung einer bestimmten Software-Version festgestellt wurden	Befunde in Bearbeitung: Anzahl von Fehlern, die vor Veröffentlichung behoben sein werden. Offene Befunde: Anzahl von Fehlern, die nach Veröffentlichung noch in der Software-Version enthalten sind.

[5] Vergleiche auch [Mock03]

Änderungen im Code	Deltas: Gesamtanzahl der Änderungen Hinzugefügt: Anzahl der hinzugefügten Code-Zeilen Gelöscht: Anzahl der gelöschten Code-Zeilen Verändert: Anzahl der veränderten Code-Zeilen
Beteiligte Personen	Autoren: Anzahl von verschiedenen Personen, die Änderungen am Code durchgeführt haben. Einfluss von unerfahrenen Autoren: Anzahl von Änderungen durch Personen, die erst wenig Erfahrung mit Änderungen im Rahmen eines bestimmten Software-Produktes haben.
Prozesseffizienz	Prozesskennzahlen (wird von [Li06] nicht verwendet)

Verwendungsmetriken: Messen die Eigenschaften der spezifischen Verwendung eines Software-Systems (siehe Tabelle 3-3)[6].

Tabelle 3-3: Verwendungsmetriken [Li06]

Prediktor	**Beschreibung**
Service Pack	Gibt an, ob es sich bei der untersuchten Software-Version um eine Major-, Minor-Release oder ein Service-Pack gehandelt hat
Monate seit Inbetriebnahme der ersten Version	Anzahl der Monate zwischen Inbetriebnahme der ersten Version und Inbetriebnahme der untersuchten Version

6 Vergleiche auch [Jone99]

Monate seit Inbetriebnahme der vorangegangenen Version	Anzahl der Monate zwischen Inbetriebnahme der untersuchten Version und Inbetriebnahme der vorangegangenen Version
Monate bis zur Inbetriebnahme der nächsten Version	Anzahl der Monate zwischen Inbetriebnahme der untersuchten Version und (geplanter) Inbetriebnahme der nachfolgenden Version

Konfigurationsmetriken: Messen die Auswirkungen verschiedener Hard- und Systemsoftware-Konfigurationen auf ein Software-Produkt (siehe auch Tabelle 3-4)[7].

Tabelle 3-4: Konfigurationsmetriken [Li06]

Prediktor	**Beschreibung**
Subsystem	Gliederung eines Software-Produktes in einzelne Funktionsgruppen
Betriebssystem	Auswirkung verschiedener Betriebssystem-Plattformen auf die Stabilität eines Software-Produktes
Version des verwendeten Internet Explorers (nur Produkt A)	Z.B. Auswirkung der verschiedenen Internet Explorer Versionen auf die Stabilität eines Software-Produktes

Die Auswahl geeigneter Prediktoren kann dabei lt. [Li06] mittels Rangkorrelation oder mit selektiven Modellierungsmethoden durchgeführt werden.

Konkret standen lt. [Li06] folgende Modellierungsmethoden zur Verfügung:

- Gleitender Durchschnitt[8]
- Exponentielle Glättung[8]
- Lineare Regression mit Modellauswahl[9]
- Clustering[10]

---

[7] Vergleiche auch [Mock05]

[8] Vergleiche auch [Li04a]

[9] Vergleiche auch [Kosh92]

[10] Vergleiche auch [Kosh93]

- Entscheidungsbäume[11]
- Neuronale Netze[12]
- Verhältnisse[13]

Die Methoden des gleitenden Durchschnitts bzw. der exponentiellen Glättung verwenden keine Prediktoren und sind daher in Hinblick auf die Fragestellung von [Li06] nach Auswahl passender Prediktoren nicht geeignet.

Clustering, Neuronale Netzwerke und Methoden der Diskriminanzanalyse vermischen die Wirkung einzelner Prediktoren und waren daher in Hinblick auf die Fragestellungen für [Li06] ebenfalls nicht geeignet. Die Verhältnis-Methode erfordert a priori die Festlegung auf einen bestimmten Prediktor und erlaubt damit ebenfalls keinen Vergleich mit anderen Prediktoren.

Lt. [Li06] lassen Entscheidungsbäume innerhalb einer bestimmten Ebene keine Rückschlüsse hinsichtlich der relativen Bedeutung einzelner Prediktoren zu. Hier widerspricht z.B. [Knab06], der sehr wohl Entscheidungsbäume einsetzt, um auch Aussagen bezüglich der Signifikanz einzelner Prediktoren zu treffen.

Lineare Regression hingegen erlaubt lt. [Li06] in Zusammenhang mit Rangkorrelation einen sehr selektiven Umgang mit einzelnen Prediktoren. Dies war für [Li06] vor allem in Zusammenhang mit der geforderten Aussagekraft und Quantifizierbarkeit der Modellierungsmethode hinsichtlich der Signifikanz einzelner Prediktoren von Bedeutung. Es ging [Li06] vor allem auch darum, geeignete komplementäre Prediktoren zu finden.

[Li06] berichtet anschließend über die erzielten Ergebnisse in den Bereichen Priorisierung des Software-Komponententests, Vorhersage der Anzahl Feldfehler im ersten Jahr nach Inbetriebnahme und über besonders signifikante Prediktoren.

Im Bereich Priorisierung des Software-Komponententests identifizierte [Li06] mittels Poisson Regression die mit den meisten Fehlern behafteten Subsysteme einer Software-Version. Die gefundenen Ergebnisse deckten sich mit Expertenaussagen bei ABB.

Im Bereich Prognose der Anzahl Feldfehler wurden von [Li06] zum Vergleich Vorhersagen mit 16 verschiedenen Modellierungsmethoden durchgeführt. Zur Überprüfung der Genauigkeit der Ergebnisse wurde eine von [Li06] als „Forward-

---

11 Vergleiche auch [Kosh02]

12 Vergleiche auch [Kosh92a] und [Kosh95]

13 Vergleiche auch [Li06a]

Prediction-Evaluation-Procedure“ bezeichnete Methode verwendet: Für die Prognoserechnung einer bestimmten Software-Version werden nur die Daten, die zum Zeitpunkt der Release verfügbar waren, verwendet. Die Feldfehlerprognose wurde dann mit der mittlerweile bekannten Anzahl an Feldfehlern verglichen. [Li06] gibt die Ergebnisse als Absolute-Relative-Error (ARE) an. Ein ARE ist definiert als

Formel 3-1: Absolute-Relative-Error (ARE) [Li06]

$$ARE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\bar{y}_i - y_i}{y_i}\right|$$

$\bar{y}_i$ = Prognostizierte Anzahl an Feldfehlern für Beobachtung i

y_i = Eigentliche Anzahl an Feldfehlern für Beobachtung i

n = Anzahl der Beobachtungen

Für das Produkt A waren mit der Methode des gleitenden Durchschnitts unter Verwendung der Daten einer Release Prognosen durchschnittlich innerhalb eines ARE von 24,6% für 3 Releases möglich (siehe Tabelle 3-5: ARE Produkt A).

Tabelle 3-5: ARE Produkt A [Li06]

ARE Produkt A	**R1.0**	**R1.1**	**R1.2**	**∅**
Gleitender Durchschnitt 1 Release	3,0%	51,7%	19,2%	24,6%
Gleitender Durchschnitt 2 Releases		54,0%	50,0%	52,0%
Gleitender Durchschnitt 3 Releases			37,8%	37,8%
Exponentielles Glätten 2 Releases		53,6%	44,6%	49,1%
Exponentielles Glätten 3 Releases			53,9%	53,9%
Lineare Regression mit Modellauswahl			17,6%	17,6%
Clustering			50,0%	50,0%
Entscheidungsbaum mit 2 Releases	3,0%	51,7%	19,2%	24,6%
Entscheidungsbaum mit 3 Releases	3,0%	54,0%	19,2%	25,4%
Entscheidungsbaum mit 4 Releases	3,0%	54,0%	62,1%	39,7%
Neuronales Netzwerk	2,3%	52,3%	53,9%	36,2%

Verhältnisse	51,4%	48,4%	22,0%	40,6%

Für Produkt B waren mit der Verhältnis-Methode Prognosen durchschnittlich innerhalb eines ARE von 77,1% für 6 Releases möglich (siehe Tabelle 3-6: ARE Produkt B).

Tabelle 3-6: ARE Produkt B [Li06]

ARE Produkt B	**R2.0**	**R3.0**	**R3.1**	**R3.2**	**R4.0**	**R4.1**	**∅**
Gleitender Durchschnitt 1 Release	61,9%	17,6%	65,3%	1628,6%	332,1%	27,3%	662,7%
Gleitender Durchschnitt 2 Releases		31,4%	71,1%	1128,6%	207,1%	20,5%	452,1%
Gleitender Durchschnitt 3 Releases		9,8%	62,0%	919,0%	154,8%	136,4%	403,4%
Exponentielles Glätten 2 Releases		29,0%	70,1%	1216,8%	229,2%	12,0%	486,0%
Exponentielles Glätten 3 Releases						410,7%	410,7%
Lineare Regression mit Modellauswahl		46,2%	70,6%	785,7%	121,4%	118,8%	342,0%
Clustering		31,4%	71,1%	564,3%	66,1%	45,5%	225,3%
Entscheidungsbaum mit 2 Releases	61,9%	17,6%	65,3%	1628,6%	332,1%	27,3%	662,7%
Entscheidungsbaum mit 3 Releases	61,9%	31,4%	71,1%	1628,6%	332,1%	27,3%	662,7%
Entscheidungsbaum mit 4 Releases	61,9%	9,8%	62,0%	1628,6%	332,1%	4,5%	655,1%
Neuronales Netzwerk	61.0%	32,4%	70,4%	1628,6%	332,1%	45,7%	668,8%
Verhältnisse	36,9%	136,7%	70,9%	160,2%	62,6%	8,4%	77,1%

[Li06] berichtet, dass eine gemeinsame Betrachtung von Service-Packs, Minor Releases und Major Releases nicht sinnvoll möglich war. Für Produkt A waren mit der Clustering-Methode bei gemeinsamer Betrachtung für 4 Releases nur mehr Prognosen innerhalb eines ARE von 64,3% möglich. Die Methode des gleitenden Durchschnitts unter Verwendung der Daten von einer Release produzierte Prognosen innerhalb eines ARE von 469,8%. Für Produkt B waren die besten Prognosen mit der Methode des gleitenden Durchschnitts unter Verwendung der Daten von drei Releases innerhalb eines ARE von 179,4% möglich. Unter Verwendung neuronaler Netzwerke konnte lediglich ein ARE von 213,8% erzielt werden.

Die schlechten Prognosen der Feldfehleranzahlen für Produkt B ergaben sich lt. [Li06] hauptsächlich wegen der Version 3.2. Diese Version hatte um 85% weniger Feldfehler als der Durchschnitt aller Versionen. Keine Modellierungsmethode konnte für diese Version korrekt vorhersagen. Solche Ausreißer verzerren die Modellparameter, wenn sie in die Bestimmung mit eingehen.

Generell stellte [Li06] fest, dass schlechte Prognosequalität auch auf eine zu geringe Anzahl an verfügbaren Beobachtungen bzw. Proben zurückzuführen ist. [Li06] standen für deren Untersuchungen lediglich 6 Releases von Produkt A und 8 Releases von Produkt B zur Verfügung.

Offen bleibt bei [Li06], inwieweit ein ARE von 24,6% (Produkt A) bzw. 77% (Produkt B) die Planung bzw. Allokation von Wartungsressourcen sinnvoll unterstützen kann.

Abschließend ermittelte [Li06] mittels Rang-Korrelation und Linearer Regression (siehe Tabelle 3-7 und Tabelle 3-8) sowie mittels Bayesschem Informationskriterium (BIC) (siehe Tabelle 3-9 und Tabelle 3-10) besonders signifikante Prediktoren.

Tabelle 3-7: Signifikante Prediktoren Produkt A [Li06]

Produkt A Prediktoren	**Korrelation**	**P-Wert**
Offene Issues	0,770	0,021
Issues in Behebung	0,803	0,015
Anzahl Service-Packs	-0,520	0,162

Tabelle 3-8: Signifikante Prediktoren Produkt B [Li06]

Produkt B Prediktoren	**Korrelation**	**P-Wert**
Monate seit Inbetriebnahme der ersten Version	-0,683	0,009

Offene Issues	-0,643	0,015
Issues in Behebung	0,424	0,130

Tabelle 3-9: Signifikante Prediktoren mittels BIC (Produkt A) [Li06]

Produkt A Prediktoren
Offene Issues
Anzahl Service-Packs

Tabelle 3-10: Signifikante Prediktoren mittels BIC (Produkt B) [Li06]

Produkt B Prediktoren
Offene Issues
Monate bis zur Inbetriebnahme der nächsten Version

Hat man lt. [Li06] einmal die besonders signifikanten Prediktoren ermittelt, kann man Maßnahmen ergreifen, um diese Prediktoren möglichst genau messen und historisch aufzeichnen zu können. Dazu benötigt man allerdings bereits eine gewisse Basis an verfügbaren Daten, was diese Festlegung am Anfang einer neuen Produktentwicklung schwierig gestalten kann. Hilfreich kann hier der Vergleich mit ähnlichen Produktentwicklungen sein, wenn dort auch Metriken zur Fehlerprognose verwendet worden sind.

Weiters stellte [Li06] abschließend fest, dass die Ergebnisse der Feldfehlerprognosen nicht nur dazu verwendet werden können, um eine Priorisierung der Testbereiche sowie eine bessere Planung der Wartungsressourcen durchführen zu können, sondern auch um vor Veröffentlichung einer neuen Software-Version das Risiko der Inbetriebnahme besser beurteilen zu können. Dies ist besonders im Bereich von „Mission Critical“ Systemen relevant.

Genauso wichtig wie die Auswahl geeigneter Prediktoren, ist lt. [Li06] also die Verwendung aussagekräftiger Modellierungsmethoden, passender Überprüfungsmethoden (wie die Forward-Prediction-Evaluation-Procedure) und der Aufbau und die Verwendung möglichst umfangreicher und lückenloser historischer Metriken-Datenreihen.

[Li06] unterstützt damit die Hypothese 1 dieser Arbeit, dass gute Feldfehlerprognosen wesentlich von der Wahl geeigneter Prediktoren abhängen. Wenn nun gewisse Metriken bzw. Prediktoren gar nicht verfügbar sind (auch wenn sie grundsätzlich geeignet wären), sind gute Prognosen ebenfalls nicht möglich.

Diese auch aus den Untersuchungen von [Li06] ableitbare Erkenntnis stützt die Hypothese 2 dieser Arbeit, dass eine eingeschränkte Verfügbarkeit von Metriken bzw. Prediktoren aufgrund nachträglicher Datengewinnung zu schlechten Prognosen führt.

3.3 Entscheidungsbaum-basiertes maschinelles Lernen

Patrick Knab, Martin Pinzger und Abraham Bernstein von der Universität Zürich berichten in [Knab06] über den Einsatz von Entscheidungsbaum-basiertem maschinellem Lernen zur Vorhersage von Feldfehlerdichten.

Dabei wurden als Prediktoren Source-Code-Metriken und Metriken zur Bestimmung des Grades der Source-Code-Veränderung verwendet. Zusätzlich untersuchte [Knab06] auch die Bedeutung von Metriken, die über mehrere Komponenten wirksame Source-Code-Veränderungen beschreiben. Die jeweils zugehörige Anzahl an Feldfehlern war natürlich auch verfügbar.

[Knab06] verwendete Entscheidungsbaum-basiertes maschinelles Lernen. [Knab06] wollte auch herausfinden, ob anhand der entstandenen Entscheidungsbäume auch Gesetzmäßigkeiten bezüglich des Zusammenhanges zwischen Prediktoren und jeweiliger Anzahl von Feldfehlern festgestellt werden können.

Auch [Knab06] verfolgte das Ziel, Projektmanagern sinnvoll einsetzbare Methoden an die Hand zu geben, um problematische Software-Komponenten möglichst früh im Software-Entwicklungsprozess feststellen und proaktiv entsprechende Ressourcen darauf konzentrieren zu können.

[Knab06] stellte folgende Hypothesen zur Diskussion:

- Hypothese 1: Die Fehlerdichte einer Release kann mit den aus derselben Release gewonnenen Metriken prognostiziert werden.
  - Hypothese 1a: Größere Source-Code-Komponenten haben eine größere Fehleranzahl als kleine Source-Code Komponenten.
  - Hypothese 1b: Größere Source-Code Komponenten haben eine höhere Fehlerdichte als kleine Source-Code Komponenten.
- Hypothese 2: Die Fehlerdichte einer Release kann unter geeigneten Bedingungen prognostiziert werden.
- Hypothese 3: Faktoren (auch Prediktoren bzw. signifikante Metriken), die zu einer höheren Fehlerdichte führen, können identifiziert werden.

- Hypothese 4: Metriken, die über mehrere Software-Komponenten wirksame Source-Code Veränderungen beschreiben, sind signifikant für die Vorhersage der Feldfehlerdichte bzw. der Feldfehleranzahl.

[Knab06] bezog sich in seiner Studie unter anderem auch auf folgende Arbeiten:

Khoshgoftaar et al. beschäftigten sich mit der Prognose von Feldfehlerdichten anhand von Code-Churn Metriken (Metriken, die z.B. die Anzahl von Veränderungen einer Software-Komponente im Lauf der Zeit messen). Software-Komponenten werden als fehleranfällig bezeichnet, wenn die Anzahl der im Rahmen des Debugging hinzugefügten bzw. veränderten Code-Zeilen ein bestimmtes Maß übersteigt. Als Methode wurde von Khoshgoftaar et al. die Diskriminanzanalyse verwendet.

Nagappan und Ball zeigten, dass relative bzw. normalisierte Code-Churn-Metriken eine gute Eignung als Prediktoren von Feldfehlerdichten, aber eine schlechte Eignung als Prediktoren von Feldfehleranzahlen haben.

Fenton et al. fanden heraus, dass die meisten während der Entwicklung festgestellten Fehler in nur wenigen Software-Komponenten enthalten sind. Auch die Anzahl der Komponenten, in denen nach Veröffentlichung die meisten Fehler enthalten sind, ist sehr klein. Dabei gibt es zwischen der Größe bzw. Komplexität dieser Software-Komponenten und der Fehleranzahl keinen unmittelbaren Zusammenhang.

Chidamer und Kemerer beschäftigten sich wie auch schon in der Studie von [Sher05] erwähnt, mit der Entwicklung von Metriken für die objektorientierte Programmierung. In Experimenten konnte nachgewiesen werden, dass sich alle gefundenen Metriken mit Ausnahme einer einzigen Metrik gut als Prediktoren zur Vorhersage der Fehlerdichte eignen.

Ostrand et al. verwendeten negative binäre Regression, um den Ort und die Anzahl von Feldfehlern in einem Software-System festzustellen. Im Rahmen dieser Untersuchungen wurde festgestellt, dass ein nur auf Größenmetriken basiertes Regressionsmodell trotz der scheinbaren Einfachheit zu guten Vorhersagen der Feldfehleranzahl führt.

Graves et al. untersuchten verschiedene statistische Modelle um herauszufinden, welche Aspekte im Änderungsverlauf einer Software-Komponente den größten Einfluss auf die Fehleranfälligkeit einer Komponente haben. Unter anderem hatte sich dabei die Anzahl aller Änderungen einer Software-Komponente zusammen mit dem Alter der Software-Komponente als geeigneter Prediktor für die Vorhersage der

Fehlerdichte erwiesen. Änderungen, die mehrere Komponenten betreffen, sind als Prediktoren wenig aussagekräftig.

Mohagheghi et al. stellten fest, dass in wiederverwendeten Komponenten weniger Fehler enthalten sind als in Komponenten, die nur in einem bestimmten Kontext verwendet werden.

[Knab06] untersuchte nun 7 Mozilla Releases aus der Zeit zwischen 2001 und 2004.

Tabelle 3-11: Untersuchte Mozilla Releases [Knab06]

#	**Release**	**Datum**
1	0.92	Juni 2001
2	0.97	Dezember 2001
3	1.0	Juni 2002
4	1.3a	Dezember 2003
5	1.4	Juni 2003
6	1.6	Jänner 2004
7	1.7	Juni 2004

In der Release 1.7 waren etwa 1300 C/C++ Source-Code- und Header-Dateien mit ungefähr 560000 LOC enthalten. Für die Experimente verwendete [Knab06] davon 366 C/C++ Source-Code-Dateien. Aus den Source-Code-Dateien wurden von [Knab06] die Metriken in Tabelle 3-12 gewonnen.

Tabelle 3-12: Metriken [Knab06]

Name	**Beschreibung**
linesOfCode	Anzahl der Code-Zeilen
nrVars	Anzahl der Variablen
nrFuncs	Anzahl Funktionen
incomingCallRels	Anzahl von externen Aufrufen von in der Komponente enthaltenen Funktionen

outgoingCallRels	Anzahl von Funktionsaufrufen in andere Komponenten
incomingVarAccessRels	Anzahl der externen Benutzung von in der Komponente enthaltenen Variablen
outgoingVarAccessRels	Anzahl der von der Komponente verwendeten externen Variablen
nrMRs	Anzahl Änderungen
sharedMRs	Anzahl Änderungen, die über mehrere Komponenten wirken
nrPRs	Anzahl Fehler
nrPRsNormal	nrPRs der Klasse „Normal“
nrPRsTrivial	nrPRs der Klasse „Trivial“
nrPRsMinor	nrPRs der Klasse „Minor“
nrPRsMajor	nrPRs der Klasse „Major“
nrPRsCritical	nrPRs der Klasse „Critical“
nrPRsBlocker	nrPRs der Klasse „Blocker“

Die Metrik sharedMRs misst also die Anzahl der zusätzlich zur betrachteten Komponente zu ändernden Software-Komponenten. Damit ist sharedMRs die Metrik, die die Wirkung von auf mehrere Komponenten wirksamen Source-Code-Veränderungen beschreibt.

Zusätzlich zu den jeweils absoluten Werten wurden von [Knab06] auch Trendwerte (Differenzen zwischen Releases) und normalisierte Werte (bezogen auf die LOC einer Komponente) berechnet. Die Normalisierung ist dabei lt. [Knab06] ganz wesentlich zur Vorhersage von Fehlerdichten.

Die Darstellung der Metriken erfolgte nach folgendem Schema:

- Das Prefix „static_“ bezeichnet absolute Metriken.
- Das Prefix „norm_“ bezeichnet normalisierte Metriken.

- Das Prefix „delta_“ bezeichnet Trendwerte.

Zusätzlich wurde die Nummer der zugehörigen Releases (siehe Tabelle 3-11) von [Knab06] am Ende jeder Metrikbezeichnung als Postfix ergänzt. Die Metrik „delta_nrMRs_4“ beschreibt also die Anzahl der Änderungsmeldungen von Release 1.3a zu Release 1.4.

Die gewonnenen Metriken wurden von [Knab06] in jeweils 5 äquidistante Klassen aufgeteilt. Die Wahrscheinlichkeit einer Instanz, einer der 5 Klassen anzugehören, war daher jeweils 20%. Die so klassifizierten Daten wurden dann unter Verwendung eines J48 Algorithmus weiter klassifiziert. J48 ist ein spezieller Algorithmus im Bereich des Entscheidungsbaum-basierten maschinellen Lernens.

[Knab06] führte sechs Experimente durch.

Experiment 1:

Verwendung aller Metriken von Release 4, ausgenommen der Fehlermetriken (nrPRs, nrPRsNormal, etc.). Ziel war die Prognose der absoluten Fehleranzahl in Release 4.

Im Experiment wurden 227 Komponenten (62%) korrekt klassifiziert, 139 Komponenten (38%) wurden falsch klassifiziert. Die zugehörige Wahrheitsmatrix zeigte, dass im Rahmen der fehlerträchtigsten Klasse sogar 82% der Komponenten richtig klassifiziert wurden. Würde man die nächstschlechtere Klasse dazu nehmen, hätte man 100% der fehlerträchtigsten Komponenten korrekt klassifiziert.

Wesentlicher Prediktor in diesem Experiment war die Anzahl der Änderungsmeldungen (static_nrMRs_4).

Experiment 2:

Verwendung aller Metriken von Release 4, ausgenommen der Fehlermetriken (analog zu Experiment 1) und zusätzlich mit Ausnahme der Änderungsmetriken (nrMRs, sharedMRs). Ziel war diesmal die Prognose der Fehlerdichte in Release 4 (norm_nrPRs_4).

Im Experiment wurden 138 Komponenten (38%) korrekt klassifiziert, 228 Komponenten (62%) wurden falsch klassifiziert. Obwohl die Ergebnisse besser waren als die initiale Aufteilung der Daten, waren sie doch wesentlich schlechter als die Ergebnisse von Experiment 1. Auch in der zugehörigen Wahrheitsmatrix waren die Ergebnisse ziemlich gleichmäßig verteilt und ließen daher keine weiteren Schlüsse zu.

Dass die deutlich schlechteren Ergebnisse durch Weglassen der Änderungsmetriken entstanden waren, ist bereits ein erstes Indiz für die Bedeutung der Änderungsmetriken als Prediktoren.

Experiment 3:

Wiederholung von Experiment 1. Diesmal war aber das Ziel die Prognose der Fehlerdichte in Release 4.

Im Experiment wurden 192 Komponenten (52%) korrekt klassifiziert, 174 Komponenten (48%) wurden falsch klassifiziert.

Die zugehörige Wahrheitsmatrix zeigte, dass entlang der Diagonale (enthält die Anzahl der korrekt klassifizierten Komponenten) unter Einbeziehung der benachbarten Fehlqualifikationen ein Großteil der untersuchten Komponenten zu finden war. Vor allem die Vorhersage bezüglich der fehlerträchtigsten Klasse war unter Einbeziehung der benachbarten Fehlqualifikationen wieder als sehr gut zu bezeichnen (sie lag bei 86%).

Experiment 4:

Verwendung aller Metriken von Release 4, mit Ausnahme der Fehlermetriken und der Metrik über verteilte Änderungen (sharedMRs, norm_sharedMRs, etc.).

Im Experiment wurden 197 Komponenten (54%) korrekt prognostiziert, 69 Komponenten (46%) wurden falsch klassifiziert. Das Ergebnis war dem von Experiment 3 ziemlich ähnlich. Es legte weiters nahe, dass die Metrik über verteilte Änderungen (also Änderungen, die mehrere Komponenten betreffen) als Prediktor nur geringe Relevanz besitzt. Die Anzahl der Änderungsmeldungen einer Komponente hängt also nicht von den logisch verknüpften Komponenten ab.

Interessanterweise enthielten die obersten beiden Ebenen des Entscheidungsbaumes von Experiment 4 aber andere Kombinationsmetriken, wie z.B. delta_norm_outgoingCallRels_4 bzw. norm_incomingCallRels_4.

[Knab06] stellte anhand der Experimente 1 bis 4 fest, dass sich die Fehlerdichte vor allem abhängig von der Anzahl der Änderungen mit einer Genauigkeit von mindestens 50% (verglichen mit der Genauigkeit von 20% im Rahmen der Basisklassifizierung anhand der Klassenvariablen) vorhersagen lässt. Hypothese 1 wurde von [Knab06] daher akzeptiert. Da LOC kein geeigneter Prediktor zur Vorhersage der Fehlerdichte ist, wurden die Hypothesen 1a und 1b von [Knab06] zurückgewiesen.

Die Experimente 5 und 6 sollten nun vor allem zur Klärung der Hypothesen 2 und 4 von [Knab06] beitragen.

Experiment 5:

Verwendung aller Metriken der Releases 3, 4, und 5 inklusive der Trendmetriken delta_nrMRs_4 und delta_norm_nrPRsMajor_5. Ziel war die Prognose der zusätzlichen Fehlermeldungen in Release 6 (delta_nrPRs_6).

Im Experiment wurden 186 Komponenten (51%) korrekt klassifiziert, 177 Komponenten (49%) wurden falsch klassifiziert. Unter Einbeziehung der benachbarten Fehlqualifikationen wurden die Klassen mit den fehlerfreisten und den fehlerträchtigsten Komponenten am genauesten klassifiziert. Für die 3 mittleren Klassen wurde eine geringere Genauigkeit beobachtet.

Ein wesentlicher Prediktor für die Vorhersage der Fehleranzahl zukünftiger Releases war die Anzahl der Fehler in historischen Releases (im konkreten Beispiel delta_nrPRsMajor_4). Metriken über verteilte Änderungen spielten in diesem Fall wieder nur eine untergeordnete Rolle.

[Knab06] stellte an dieser Stelle auch fest, dass keine einfachen bzw. grundsätzlich gültigen Zusammenhänge zwischen Prediktoren und der prognostizierten Fehlerdichte bestehen.

Experiment 6:

Wiederholung von Experiment 5, nur wurden im Experiment 6 die zusätzliche Fehleranzahl für Release 7 anhand der Releases 3 bis 6 prognostiziert.

Im Experiment wurden 215 Komponenten (59%) korrekt klassifiziert, 148 Komponenten (41%) wurden falsch klassifiziert. Damit waren die Ergebnisse von Experiment 6 besser als die von Experiment 5. Interessanterweise war im Fall des Experimentes 6 die Anzahl der Funktionen (static_nrFuncs_6) der relevanteste Prediktor. Aber auf der zweiten Ebene des Entscheidungsbaumes befanden sich bereits die Fehlermetriken der historischen Releases.

[Knab06] versuchte mit den Metriken aus Experiment 6 auch die normalisierte Anzahl der zusätzlichen Fehler zu prognostizieren. Dabei verschlechterte sich das Ergebnis aber gravierend. Es wurden nun nur mehr 162 Komponenten (45%) korrekt klassifiziert, 201 Komponenten (55%) wurden falsch klassifiziert.

Auf Basis der Ergebnisse der Experimente 4 und 5 stellte [Knab06] fest, dass es tatsächlich möglich ist, zukünftige Fehlerdichten zu prognostizieren. Hypothese 2 wurde von [Knab06] daher akzeptiert. Hypothese 3 musste von [Knab06] allerdings

zurückgewiesen werden, da trotz der Verwendung von Entscheidungsbäumen keine grundlegenden Zusammenhänge zwischen Metriken bzw. Prediktoren und der prognostizierten Fehlerdichte gefunden werden konnten. Hypothese 4 wurde von [Knab06] ebenfalls zurückgewiesen, da Metriken über die Anzahl von verteilten Änderungen weder für die Vorhersage von Fehlerdichten bzw. Fehleranzahlen innerhalb einer Release noch für die Vorhersage von Fehlerdichten bzw. Fehleranzahlen zukünftiger Releases besonders relevant waren.

Zusammenfassend stellte [Knab06] fest, dass einfaches Entscheidungsbaum-basiertes maschinelles Lernen grundsätzlich gute Resultate im Rahmen der Vorhersage bzw. Klassifizierung von Fehlerdichten und Fehleranzahlen liefert. LOC ist ein geeigneter Prediktor für die Vorhersage von Fehleranzahlen, für die Prognose von Fehlerdichten eignet er sich nicht. Historische Daten über Fehlerdichten bzw. Fehleranzahlen sind für die Prognose von zukünftigen Fehlerdichten bzw. Fehleranzahlen von großer Bedeutung.

Weitere Erkenntnisse von [Knab06] waren, dass die Anzahl der Fehlermeldungen stark linear mit LOC korreliert ist, d.h. die Fehlerdichte ist in großen Komponenten nicht grundsätzlich höher als in kleinen.

Auch [Knab06] stützt mit seinen Ergebnissen damit alle 3 Hypothesen dieser Arbeit.

3.4 Nearest Neighbor Sampling

Gary D. Boetticher von der University of Houston, Texas berichtet in [Boet05] darüber, wie man durch gezielte Auswahl von Datensätzen (Trainingsdatensätzen) im Rahmen der Modellerstellung zu besseren Modellen für die Feldfehlerprognose kommt. [Boet05] möchte damit auch die Akzeptanz dieser Methoden bei Projektmanagern erhöhen und damit den Einsatz solcher Methoden vorantreiben.

[Boet05] stellte fest, dass ein wesentlicher Aspekt bei der Erstellung von Vorhersagemodellen die geeignete Auswahl von Trainingsdatensätzen ist. Üblicherweise basieren Auswahlverfahren auf dem Klassenattribut (Anzahl Feldfehler) anstatt der Nicht-Klassenattribute (zugeordnete Metriken). Wenn aber der nächste Nachbar (bezogen auf die Nicht-Klassenattribute) eines Testdatensatzes im Trainingsdaten-Set von der gegenüberliegenden Klasse stammt, dann ist eine schlechte Prognose (Fehlklassifizierung) sehr wahrscheinlich.

Um die Auswirkungen zu zeigen, die Nicht-Klassenattribute auf die Erstellung von Vorhersagemodellen haben, experimentierte [Boet05] mit 5 public domain Fehlerdaten-Sets aus dem NASA Metrics Data Program (MDP) und dem PROMISE

Repository (http://promise.site.uoittawa.ca/SERepository). Die 5 Fehlerdaten-Sets CM1, JM1, KC1, KC2 und PC1 bestehen aus statischen Code-Metriken und einer zugeordneten Fehlerrate (Klassenattribut).

Tabelle 3-13: Daten-Sets aus dem NASA Metrics Data Program (MDP) [Boet05]

Projekt	**Source-Code**	**Beschreibung**
CM1	C	NASA Raumfahrtsinstrument
KC1	C++	Speichermanagement-Software
KC2	C++	Verarbeitung wissenschaftlicher Daten
JM1	C	Echtzeitsystem
PC1	C	Flug-Software eines Erdsatelliten

Jedes Fehlerdaten-Set besteht aus 21 Software-Produkt-basierten Metriken über die Größe, die Komplexität und das verwendete Vokabular. Die Größen-Metriken bestehen aus:

- Total LOC
- Ausführbare LOC
- Kommentar LOC
- Leere LOC
- LOC mit ausführbarem Code und Kommentar
- Anzahl Verzweigungen

Die Komplexitäts-Metriken bestehen aus:

- Zyklomatischer Komplexität
- Wesentlicher Komplexität
- Modul-Design Komplexität

Die lexikalischen-Metriken bestehen aus:

- Halstead-Länge
- Halstead-Volumen
- Halstead-Level

- Halstead-Difficulty
- Halstead-Intelligent-Content
- Halstead-Programming-Effort
- Halstead-Error-Estimate
- Halstead-Programming-Time
- Anzahl eindeutiger Operatoren
- Anzahl eindeutiger Operanden
- Gesamtzahl Operatoren
- Gesamtzahl Operanden

Im MDP-Fehlerdaten-Set sind numerische Werte für die Fehlerrate enthalten. Im PROMISE-Fehlerdaten-Set sind die numerischen Werte den Klassen TRUE (Komponente enthält 1 oder mehrere Fehler) und FALSE (Komponente enthält keinen Fehler) zugeordnet.

Die Fehlerdaten-Sets wurden von [Boet05] zunächst gefiltert (doppelte Datensätze und Fehlerdaten-Sets mit fragwürdigen Werten entfernt). [Boet05] verwendete nun die Fehlerdaten-Sets aus Tabelle 3-14 für seine Experimente.

Tabelle 3-14: Gefilterte Fehlerdaten-Sets [Boet05]

Projekt	**Original Anzahl Komp.**	**Anzahl Komp. nach Filterung**	**0 Fehler**	**1+ Fehler**	**% Fehler**
CM1	498	441	393	48	10,9%
JM1	10885	8911	6904	2007	22,5%
KC1	2109	1211	896	315	26,0%
KC2	522	274	269	105	28,1%
PC1	1109	953	883	70	7,3%

[Boet05] führte nun je Projekt 20 Experimente durch. Für jedes Experiment wurden 40% des Gesamtdatenbestandes als Trainings-Daten-Set verwendet, der Rest als Testdaten-Set. Im Rahmen der Aufteilung wurde von [Boet05] sichergestellt, dass das Verhältnis von Fehlern zu Anzahl Komponenten im jeweiligen Daten-Set dem Verhältnis im Gesamtdatenbestand entspricht.

Dann wurden von [Boet05] alle Nicht-Klassenattribute durch Division mit ($Maximum_k$ - $Minimum_k$) (k entspricht der jeweiligen „Spalte“ der Nicht-Klassenattribute) normalisiert. Diese Vorgangsweise stellte sicher, dass alle „Spalten“ der Nicht-Klassenattribute gleich gewichtet waren.

Anschließend wurde jeder Testdatensatz mit jedem Trainingsdatensatz verglichen und die minimale euklidische Distanz zwischen den einzelnen Datensatzpaaren berechnet. Datensatzpaare mit minimaler euklidischer Distanz und gleichem Klassenwert wurden den „Nice“ Testdaten-Sets zugeordnet, Datensatzpaare mit minimaler euklidischer Distanz und gegenüberliegendem Klassenwert wurden den „Nasty“ Testdaten-Sets zugeordnet. Insgesamt stellte [Boet05] so 150 Daten-Sets zusammen (5 Projekte, 20 Experimente je Projekt, 1 Trainings- und 2 Testdaten-Sets je Experiment).

Zur Berechnung von Feldfehlerprognosen mit den jeweiligen Testdaten-Sätzen wurden Decision-Tree-Learner in Zusammenhang mit dem Public Domain Data-Mining-Tool WEKA herangezogen. [Boet05] verwendete für seine Experimente Naive Bayessche und J48 Learner (den auch [Knab06] für seine Experimente verwendet hat).

Zur Auswertung der Experimente wurden von [Boet05] drei Metriken verwendet:

- PD: Wahrscheinlichkeit, eine fehlerbehaftete Komponente korrekt zu identifizieren (Wahrscheinlichkeit, eine fehlerbehaftete Komponente korrekt zu identifizieren dividiert durch die Gesamtanzahl aller fehlerhaften Komponenten)
- PF: Wahrscheinlichkeit eines falschen Alarms (Wahrscheinlichkeit, eine fehlerfreie Komponente als fehlerhaft zu klassifizieren dividiert durch die Gesamtanzahl aller fehlerfreien Komponenten)
- Acc: Genauigkeit (Wahrscheinlichkeit, fehlerbehaftete und fehlerfreie Komponenten korrekt zu identifizieren dividiert durch die Gesamtanzahl aller Komponenten)

Tabelle 3-15: Exemplarische Wahrheitsmatrix [Boet05]

	Fehlerhafte Komponente gefunden	**Fehlerfreie Komponente gefunden**
Fehlerhafte Komponente vorhanden	A Vorhersage = WAHR Realität = WAHR	B Vorhersage = FALSCH Realität = WAHR

Fehlerfreie Komponente vorhanden	C Vorhersage = WAHR Realität = FALSCH	D Vorhersage = FALSCH Realität = FALSCH

Anhand der Wahrheitsmatrix in Tabelle 3-15 stellte [Boet05] diese Metriken durch einfache Gleichungen dar.

Formel 3-2: Metriken [Boet05]

$$PD = A / (A+B)$$

$$PF = C / (C+D)$$

$$Acc = (A+D) / (A+B+C+D)$$

[Boet05] erzielte im Rahmen der Experimente folgende Ergebnisse:

Die „Nice“ Testdaten-Sets lieferten sehr gute Prognosen, durchschnittlich 94% Genauigkeit (Acc). Die „Nasty“ Testdaten-Sets lieferten entsprechend schlechte Prognosen, durchschnittlich 20% Genauigkeit (Acc). Das JM1 Daten-Set mit einer wesentlich größeren Anzahl an Komponenten als alle anderen Daten-Sets lieferte im Bereich der „Nice“ Testdaten-Sets überdurchschnittlich gute Prognosen, während es im Bereich der „Nasty“ Testdaten-Sets überdurchschnittlich schlechte Prognosen lieferte.

Wesentliche Erkenntnis von [Boet05] war, dass „Nice“ Testdaten-Sets dazu tendieren, eher fehlerfreie Komponenten vorherzusagen, während „Nasty“ Testdaten-Sets dazu tendieren, eher fehlerhafte Komponenten vorherzusagen. Diese Erkenntnis konnte [Boet05] im Rahmen dieser Studie auch mit einem weiteren Experiment unter Verwendung des KC1 Daten-Sets erhärten.

[Boet05] schlug auf Basis seiner Erkenntnisse zwei Anwendungen vor:

Man ermittelt die Datensatzpaare zwischen Trainings- und Testdaten-Set mit dem geringsten euklidischen Abstand. Hat so ein Paar den gleichen Klassenwert, so wird ein Zähler um 1 erhöht. Man kann dann eine Metrik namens experimentelle Komplexität (Exp_Difficulty) wie folgt definieren:

Formel 3-3: Experimentelle Komplexität [Boet05]

$$Exp\_Difficulty = 1 – (Zähler / Gesamtanzahl\ Testdatensätze)$$

Diese experimentelle Komplexität macht eine qualitative Aussage darüber, wie schwierig es ist, bei gegebenen Fehlerdaten-Sets ein gutes Modell herzustellen.

Ein anderer Ansatz beschäftigt sich mit dem gesamten Datenbestand, ohne ihn vorher in ein Trainings- und Testdaten-Set aufzuteilen. Man bildet wieder alle Paare mit dem geringsten euklidischen Abstand bezogen auf die Nicht-Klassenattribute. Hat so ein Paar den gleichen Klassenwert, wird wieder ein Zähler erhöht. Analog zur experimentellen Komplexität kann man die Gesamtkomplexität (Overall_Difficulty) wie folgt definieren:

Formel 3-4: Gesamtkomplexität [Boet05]

$$\text{Overall\_Difficulty} = 1 - (\text{Zähler} / \text{Gesamtanzahl Datensätze})$$

Die Gesamtkomplexität dient z.B. als Anhaltspunkt, ob es sinnvoll ist, ein Daten-Set aufzuteilen, um die experimentelle Komplexität zu verringern und damit bessere Modelle herstellen zu können.

[Boet05] liefert keine direkten Argumente für oder gegen die in dieser Arbeit im Kapitel Ziel der Arbeit aufgestellten Hypothesen. Das „Nearest Neighbor Sampling“ macht aber auch nur Sinn, wenn entsprechend umfangreiche Datenreihen aus Klassen- und Nicht-Klassenattributen verfügbar sind. Damit stützt [Boet05] zumindest indirekt die Hypothesen dieser Arbeit.

3.5 Abschätzung von Fehlerdichten

Mark Sherriff, Laurie Williams und Mladen Vouk von der North Carolina State University in Raleigh sowie Nachiappan Nagappan von Microsoft Research in Redmond berichten in [Sher05] über Erfahrungen in Zusammenhang mit der frühzeitigen Abschätzung von Fehlerdichten unter Verwendung eines im Entwicklungsprozess integrierten Haskell-Metrics-Modells.

Ziel von [Sher05] war die optimale Planung und Durchführung von Fehlerkorrekturen im Rahmen des Softwareentwicklungsprozesses. Je früher Fehler erkannt und behoben werden können, desto günstiger ist die Fehlerbehebung und desto besser die resultierende Software-Qualität. Direkt aus dem Entwicklungsprozess heraus sollten möglichst einfache Metriken gewonnen werden, die schon während der noch laufenden Entwicklung eine Einschätzung der Fehlerdichte erlauben. [Sher05] verwendete in diesem Zusammenhang die Fehlerdichte einer Software-Komponente als extern sichtbares Qualitätskriterium und verweist auf die Meinung des ISO/IEC-Gremiums, dass Metriken wertlos sind, solange sie nicht mit extern sichtbaren Qualitätskriterien verknüpft werden.

[Sher05] bestätigt, dass Software-Metriken Indikatoren für die Software-Qualität sein können. Strukturelle OO-Metriken wie die aus der Chidamber-Kemerer (CK)

Metrics-Suite wurden bereits im Jahre 1994 zur Untersuchung und Vorhersage von Software-Fehlern entwickelt und verwendet. Die CK-Metrics-Suite besteht dabei aus den Metriken:

- Anzahl Methoden je Klasse (gewichtet) (WMC)
- Grad der Abhängigkeit zwischen einzelnen Objekten (CBO)
- Tiefe des Vererbungsbaumes (DIT)
- Anzahl der Vererbungen einer Klasse (NOC)
- Response for a class (RFC)
- Unabhängigkeit der Methoden einer Klasse (LCOM)

Die CK-Metriken wurden in verschiedenen Studien verwendet. Dabei hat sich herausgestellt, dass je nach Studie die verschiedenen CK-Metriken unterschiedlich stark mit der Anzahl der enthaltenen Fehler korreliert waren.

[Sher05] unterscheidet zwischen strukturellen Metriken wie LOC, Anzahl der Knoten bzw. Schleifen in einem Flussdiagramm und Prozess-Metriken. Zu den Prozess-Metriken zählt [Sher05] z.B. Anzahl der Fehler, Fehlerintensität (Anzahl Fehler je Testfall), Häufigkeit von Änderungen oder Nutzungsintensität bestimmter Komponenten.

Die Untersuchungen von [Sher05] basierten auf einer Haskell-Entwicklungsumgebung. Haskell ist eine funktionsorientierte Programmiersprache. Zur Qualitätssicherung wurden lt. [Sher05] folgende Tools eingesetzt:

- HUnit: Dabei handelt es sich um ein Open Source Modultest-Framework, das speziell für Haskell Systeme entwickelt wurde. HUnit unterstützt dabei die Verwendung des sogenannten Test-Driven-Development (TDD): Zuerst überlegen sich Entwickler Testfälle zu den Anforderungen und schreiben erst danach das zugehörige Programm. TDD ist eine sehr effektive Entwicklungsmethode. In einer z.B. bei IBM durchgeführten Studie, konnte durch den Einsatz von TDD die Fehlerdichte um 40% gesenkt werden.
- QuickCheck ist ebenfalls ein Haskell-spezifisches Test-Tool, das zum halbautomatischen Testen geeignet ist.

Die Metriken wurden von [Sher05] mittels STREW-H ermittelt. STREW-H basiert auf der Java-Testsuite STREW-J.

STREW steht für “Software Testing and Reliability Early Warning” Metric-Suite und ermittelt Metriken über die Art und Weise der eingesetzten White-Box

Testverfahren und einiger struktureller Aspekte des implementierten Codes. Die in STREW verwendeten Metriken prüfen sich dabei gegenseitig auf Plausibilität. STREW liefert auf Basis der ermittelten Metriken auch eine Abschätzung der Fehlerdichte. Als Verfahren kommt dabei eine multiple lineare Regression zum Einsatz. Die Signifikanz einzelner Metriken (Prediktoren) für die Abschätzung der Feldfehlerdichte wurde mittels Bayesschem Informationskriterium und Hauptkomponentenanalyse (englisch Principal Component Analyses bzw. PCA) festgestellt.

Auf Basis einer umfangreichen Tauglichkeitsstudie mit einem 200+ KLOC Open-Source Haskell-Projekt wurden von [Sher05] 5 Metriken für die Verwendung mit STREW-H identifiziert:

- Anzahl LOC der Test-Source-Dateien: Ist ein Maß für die Testintensität relativ zur Systemgröße.
- Anzahl Type-Signatures pro Anzahl Methoden: Ist ein Maß für den Anteil von Type-Signatures am gesamten Code. Type-Signatures stellen unter Haskell eine gute Programmierpraxis dar.
- Anzahl Testfälle pro Anzahl Anforderungen: Ist ein Maß für die Testintensität gemessen am Funktionsumfang des Systems.
- Pattern-Warnings pro KLOC: Pattern-Warnings sind Compiler-Warnungen des Haskell-Compilers und beziehen sich auf schlecht programmierte Programmteile. Diese Metrik stellt also ein Maß für die Qualität des Source-Codes dar.
- Monadic-Instances of Code pro KLOC: Die Verwendung von Monadic-Instances stellt unter Haskell ebenfalls keine gute Programmierpraxis dar. Diese Metrik ist daher ebenfalls ein Maß für die Qualität des Source-Codes.

Ein interessanter Aspekt ist hier, dass [Sher05] auch Compiler-Warnungen als Metriken und damit als Prediktoren heranzieht.

Die Untersuchungen von [Sher05] wurden im Rahmen der 7-monatigen Entwicklung eines ASN.1 Compiler-Systems durchgeführt. Die etwa 20 KLOC große Entwicklung sollte auch zeigen, dass durch Verwendung funktioneller Programmiersprachen hochsichere und hochverlässliche Software-Systeme gebaut werden können.

Während der Entwicklung wurden von [Sher05] in ein- bis zweiwöchigen Intervallen 20 Momentaufnahmen der Fehlerdichte gemacht. Die Fehler wurden dabei nicht klassifiziert. Die Analyse der Daten wurde erst im Nachhinein

durchgeführt. [Sher05] empfiehlt aber, die beschriebene Methode direkt im Rahmen des Entwicklungsprozesses einzusetzen, um möglichst früh quantifizierbare Aussagen über die Source-Code-Qualität machen zu können. Diese Aussagen können dann als Entscheidungsgrundlage für Anpassungen im Entwicklungsprozess verwendet werden.

[Sher05] analysierte die Daten unter Verwendung der STREW-H Metriken und der in den Momentaufnahmen enthaltenen Fehlerdichten. Das multiple lineare Regressionsmodell wurde mit 14 Momentaufnahmen erstellt und mit den restlichen 6 Momentaufnahmen überprüft. Um die möglichen Auswertungsfehler zu minimieren, wurde die Regression fünfmal mit einer jeweils anderen Aufteilung der Momentaufnahmen gerechnet.

Die R^2-Werte der 5 Berechnungen waren wie folgt:

0.943, 0.930, 0.962, 0.949 und 0.967

Alle R^2-Werte lagen also nahe bei 1 und belegten damit die gute Übereinstimmung des Modells (der gewählten Metriken bzw. Prediktoren) mit der Realität.

Zur weiteren Betrachtung wurde von [Sher05] das Modell mit dem kleinsten R^2-Wert herangezogen. 5 von 6 Prognosen mit diesem Modell waren innerhalb eines Fehlers von 0.3 Fehlern/KLOC, eine Prognose war allerdings bis zu 0.8 Fehler/KLOC vom realen Wert entfernt (die dieser Prognose zugrundeliegende Momentaufnahme wurde offenbar zu einer Zeit gemacht, in der verstärkte Testaktivitäten gelaufen sind).

Statistische Analysen des Ergebnisses durch [Sher05] zeigten außerdem, dass die Kombination von Metriken eine höhere Signifikanz aufweisen kann als einzelne Metriken. Es hat sich gezeigt, dass im konkreten Fall die Metrik Pattern-Warnings/KLOC besonders signifikant war.

[Sher05] bestätigt also ebenfalls die Hypothese 1 dieser Arbeit. Auch Hypothese 3, dass die Rahmenbedingungen für Feldfehlerprognosen bereits zu Beginn der Software-Entwicklung festgelegt werden müssen, um Feldfehlerprognosen hoher Qualität zu erhalten, wird durch die Vorgangsweise von [Sher05] bestätigt.

3.6 Prognose von Feldfehlerraten

Paul Luo Li, James Herbsleb und Mary Shaw vom Institute for Software Research, International Carnegie Mellon University, Pittsburgh. PA sowie Bonnie Ray und P. Santhanam vom Center for Software Engineering IBM T.J. Watson Research Center, Hawthorne NY berichten in [Li04] über Ansätze und Erfahrungen in Zusammenhang

mit der Bestimmung von Feldfehlerraten für weitverbreitete Software-Produkte (z.B. eine PC-Textverarbeitung, einen Internet-Explorer, etc.).

[Li04] untersuchte dazu folgende Fragestellungen:

- Wann kann man ein für den weit verbreiteten Verkauf bestimmtes Software-Produkt in den Verkauf geben?
- Ist das Risiko bzgl. möglicher noch enthaltener Feldfehler beherrschbar?
- Ist dieses Risiko versicherbar?
- Welche Art und Menge an Ressourcen muss man für die Produktwartung bereithalten?

[Li04] versuchte Antworten auf diese Fragen zu finden, indem Methoden untersucht wurden, mit denen die Feldfehlerraten verschiedener weit verbreiteter Software-Produkte je Zeiteinheit vorherzusagen. [Li04] suchte Modelle, die die auftretenden Feldfehlermuster genau genug vorhersagen bzw. projizieren konnten. Weiters untersuchte [Li04] die Frage, ob die gefundenen Modelle durch Anpassung der Modell-parameter mit naiven Extrapolationsmethoden auch für spätere Versionen verwendet werden konnten.

Untersucht wurden folgende weit verbreitete Software-Produkte:

- 8 Releases eines kommerziellen Betriebssystems
- 8 Releases eines Open Source Betriebssystems (OpenBSD)
- 3 Releases eines kommerziellen Middleware-Systems
- 3 Releases eines Open Source Middleware-Systems (Tomcat)

Ein weitverbreitetes Software-Produkt ist dabei lt. [Li04] von folgenden Parametern bestimmt:

- Multi-Release: das Produkt wird ständig erweitert bzw. verbessert und steht deshalb in verschiedenen Releases zur Verfügung.
- Multi-Plattform: das Produkt läuft auf verschiedenen HW/SW-Plattformen.
- Weit verbreitet: über verschiedene Vertriebskanäle bzw. über das Internet befinden sich solche Software-Produkte weltweit im Einsatz.
- Die jeweiligen Umgebungs- und Verwendungsbedingungen sind dabei im Detail nicht bekannt.

Im Bereich der kommerziellen Systeme gestaltete sich die Gewinnung von Fehlerdaten für [Li04] sehr günstig. Vom kommerziellen Betriebssystem waren

eindeutig zuordenbare Fehlermeldungen verfügbar. Auf Grund der zeitlichen Struktur der verfügbaren Fehlerdaten wurde von [Li04] als Betrachtungszeitraum das Kalenderquartal gewählt. Die Fehlerdaten des kommerziellen Middleware-Systems enthielten potentiell doppelte Fehlermeldungen, der gewählte Betrachtungszeitraum war ein Monat.

Für die Open Source Produkte waren zwar ausreichend Fehlerdaten verfügbar, es lag aber in der Natur dieser Produkte, dass die zugehörigen Fehlerdaten auch Falsch- bzw. Doppelmeldungen enthielten. Als Betrachtungszeitraum für Open Source Produkte wurde von [Li04] generell das Kalendermonat gewählt.

[Li04] definierte einen Feldfehler als ein vom Benutzer gemeldetes Problem, das die Intervention eines Entwicklers zur Korrektur des Problems erforderlich machte.

Das Erscheinungsmuster von Feldfehlern bzw. sogenannte Feldfehlermuster sind dabei lt. [Li04] durch die Anzahl von auftretenden Feldfehlern je Zeiteinheit über die gesamte Lebensdauer einer Software-Release bestimmt. Die Lebenszeit einer Release ist lt. [Li04] als Zeit zwischen Verkaufsbeginn bzw. Verfügbarkeitsbeginn (Open Source Produkte) und dem Zeitpunkt, zu dem nach 3 aufeinanderfolgenden Beobachtungszyklen keine Feldfehler mehr gemeldet werden, bestimmt.

Eine wesentliche Eigenschaft von weit verbreiteten Software-Systemen ist, dass die Bestimmung von Feldfehlermustern ohne genaue Kenntnis der jeweiligen Umgebungs- bzw. Verwendungsbedingungen erfolgen muss. Es müssen aber trotzdem Eigenschaften gefunden werden, die die Feldfehlermuster beeinflussen. [Li04] definierte die wesentlichen Eigenschaften dabei wie folgt:

- Inhalt: durch ständige Verbesserungen und Weiterentwicklungen verändert sich der Inhalt von Software-Produkten mit der Zeit. Verschiedene Änderungen besitzen dabei verschiedene Feldfehlermuster.
- Entwicklungsprozess: Obwohl sich der Entwicklungsprozess in vielen Unternehmen mit der Zeit nur langsam ändert, hat er vor allem in Zusammenhang mit anderen Faktoren einen Einfluss auf das resultierende Feldfehlermuster. Zeitdruck sowie sich verändernde Entwicklungsteams und –umgebungen können zu schlechterer Produktqualität führen, die sich in einer größeren Feldfehleranzahl bzw. im sogenannten Fehler-Blocking äußert. Fehler-Blocking bedeutet, dass bestimmte im Software-Produkt enthaltene Fehler das Auftreten von anderen Fehlern zunächst verhindern. Nach der Beseitigung des einen Fehlers treten danach andere auf.

- Art der Verwendung: Verschiedene Benutzer verwenden nicht die für ihren Verwendungszweck optimale Release, andere Benutzer verwenden nicht die aktuellste Release. Wieder andere Benutzer überspringen Releases (z.B. Patch-Releases) und wechseln erst zu neuen Releases, wenn damit auch die Verfügbarkeit eines größeren Funktionsumfanges verbunden ist. Auch die Ausnutzung der gebotenen Funktionalität einer Release kann zwischen Benutzern schwanken. Je intensiver ein System genutzt wird, desto mehr Feldfehler können auftreten.
- Verwendete SW/HW-Plattform: Da nicht alle möglichen SW/HW-Plattformen, auf denen ein weit verbreitetes Software-Produkt laufen kann, im Vorhinein im Detail getestet werden können, sind bestimmte SW/HW-Plattformen fehleranfälliger als andere.

Diese Eigenschaften bestimmen also lt. [Li04] das Fehlermodell und die Möglichkeit der Modellanpassung durch Parameterextrapolation für spätere Software-Releases.

[Li04] stellte nun folgende Hypothesen auf:

- Das am besten geeignete Modell zur Vorhersage von Feldfehlermustern ist das Weibull-Modell.
- Naive Extrapolationsmethoden eignen sich nicht zur Anpassung der Modellparameter für künftige Releases.

Um diese Hypothesen zu stützen, verglich [Li04] zunächst die Modelle aus Tabelle 3-16: Modelle zur Feldfehlerratenprognose miteinander.

Tabelle 3-16: Modelle zur Feldfehlerratenprognose [Li04]

Modelltyp	**Modellname**	**Modellform**	**Modellauthoren**
Exponentialmodell	Nicht-homogenes Poisson Modell	$\lambda(t) = N\alpha e^{-\alpha t}$	Goel & Okumoto
Weibull	Weibull	$\lambda(t) = N\alpha\beta t^{\alpha-1} e^{-\beta t^{\alpha}}$	Schick-Wolverton
Gamma	S-Form Reliability-Growth-Model	$\lambda(t) = N\beta^{\alpha} t^{\alpha-1} e^{-\beta t}$	Yamada, Ohba & Osaki
Power	Duane Modell	$\lambda(t) = \alpha\beta e^{-\beta t}$	Duane
Logarithmisches Modell	Musa-Okumoto logarithmisches Poisson-Modell	$\lambda(t) = \alpha(\alpha\beta t + 1)^{-1}$	Musa-Okumoto

Alle Modelle sind entsprechend parametrierbar. Das von [Li04] favorisierte Weibull-Modell hat drei Parameter: N, α und β. Das Weibull-Modell kann daher in drei Teile zerlegt werden.

N repräsentiert die Gesamtanzahl aller Feldfehler während der Lebenszeit einer Release.

$\alpha\beta t^{\alpha-1}$ stellt einen ansteigenden Term dar, der am Beginn des Lebenszyklus einer Release dominiert.

$e^{-\beta t^{\alpha}}$ stellt einen absteigenden Term dar, der am Ende des Lebenszyklus einer Release dominiert.

N kann für verschiedene Versionen einer Software verschieden sein, abhängig von Änderungen des Release-Inhaltes bzw. des Entwicklungs-Prozesses. Mit den beiden Termen können sowohl konkave als auch konvexe Zusammenhänge dargestellt werden.

Konkave Feldfehlermuster zu Beginn des Lebenszyklus einer Software-Release treten lt. [Li04] auf, wenn zu Beginn des Lebenszyklus einer Release besonders viele Feldfehler auftreten, z.B. wenn viele Benutzer gleichzeitig viele Funktionen einer neuen Release verwenden. Konvexe Feldfehlermuster zu Beginn des Lebenszyklus einer Software-Release treten lt. [Li04] auf, wenn Benutzer erst langsam mit dem Einsatz einer neuen Software-Release beginnen bzw. wenn Probleme im Entwicklungsprozess Fehler-Blocking begünstigen (viele Fehler treten dadurch erst zu einem späteren Zeitpunkt auf).

Konkave Feldfehlermuster am Ende des Lebenszyklus einer Software-Release treten lt. [Li04] auf, wenn die Feldfehlerraten bis kurz vor dem Ende des Lebenszyklus hoch bleiben. Dies kann z.B. auf Probleme mit dem Entwicklungsprozess oder dem Inhalt der Software-Release zusammenhängen. Konvexe Feldfehlermuster am Ende des Lebenszyklus einer Software-Release treten lt. [Li04] auf, wenn Benutzer am Ende der Lebenszyklus rasch zu einer neueren Release des Software-Produktes wechseln.

Es schien für [Li04] daher naheliegend, dass ein Modell mit einem ansteigenden und einem fallenden Term, das jeweils sowohl konvexe als auch konkave Zusammenhänge modellieren kann, besser geeignet ist als ein Modell, dem eine dieser Eigenschaften fehlt. Im Vergleich zum Weibull-Modell besitzen beispielsweise das exponentielle Modell, das Power-Modell und das logarithmische Modell nicht gleichzeitig ansteigende sowie fallende Terme. Sie können daher nicht das Zusammenspiel zwischen ansteigenden und fallenden Trends im

Feldfehlermuster darstellen. Das Gamma-Modell hat zwar sowohl einen ansteigenden als auch einen fallenden Term, der fallende Term kann aber nur konvexe Sachverhalte darstellen.

Unter Verwendung einer nicht linearen Kleinsten-Quadrate-Regressions-Methode bestimmte [Li04] nun die Modellparameter für die verschiedenen Releases. Zum Vergleich der Ergebnisse wurde von [Li04] das Akaike Informations-Kriterium verwendet.

Formel 3-5: Akaike Informations-Kriterium [Li04]

$$AIC = \ln \sigma^2 + \frac{M}{T} 2$$

Mit σ^2 als Quadrat der Varianz der Residuen, M der Anzahl der Parameter und T der Anzahl der beobachteten Stichprobenwerte. Je kleiner der Wert für AIC ausfällt, umso besser spiegelt das Modell die Wirklichkeit wieder.

Bei 16 der verglichenen 22 Releases (das entspricht 73% aller betrachteten Releases) hatte das Weibull-Modell die besten AIC-Werte. Das nächstbeste Modell, das Gamma-Modell hatte nur bei 8 der 22 verglichenen Releases (das entspricht nur 36% aller betrachteten Releases) einen höheren AIC-Wert als das Weibull-Modell. Die AIC-Werte des Weibull-Modells lagen mit Ausnahme einer einzigen Release immer in einem 95%-Konfidenzintervall rund um den jeweils besten AIC-Wert. Dieses Ergebnis bestätigte daher die Hypothese von [Li04], dass das Weibull-Modell das am besten geeignete Modell zur Modellierung von Feldfehlermustern darstellt.

[Li04] validierte nun die Weibull-Modelle der verschiedenen Releases unter Verwendung der Methode der Theil-Statistik. Die Theil-Statistik vergleicht jeden Modellwert für ein gegebenes Zeitintervall i mit der statischen „Prognose“ (Verwendung des Wertes) des vorangegangenen Zeitintervalls.

Formel 3-6: Theil-Statistik [Li04]

$$U^2 = \frac{\sum (P_i - A_i)^2}{\sum A_i^2}$$

P_i prognostizierte Änderung zum Zeitintervall i-1

A_i eigentliche Änderung zum Zeitintervall i-1

U ist größer oder gleich 0.

Eine Theil-Statistik von 0 ergibt sich daher für vollständig richtige Prognosen. Eine Theil-Statistik von 1 ergibt sich für Modelle, die nicht besser als die statische

Prognose sind. Wert größer als 1 ergeben sich für Modelle, die schlechter als die statische Prognose sind.

[Li04] stellte fest, dass alle berechneten Weibull-Modelle besser waren als die statische Prognose.

Um nun auf Basis eines erstellten Weibull-Modells auch Aussagen über künftige Releases machen zu können, untersuchte [Li04] Möglichkeiten der Extrapolation der Modellparameter. Dabei wurden die Methoden des gleitenden Durchschnitts und der exponentiellen Glättung betrachtet. Da beide Methoden keine Rücksicht auf Änderungen in den vier Basiseigenschaften von weit verbreiteten Software-Produkten nehmen, nahm [Li04] an, dass durch Verwendung dieser Methoden keine wirklich brauchbaren Ergebnisse geliefert werden.

Die Ergebnisse bestätigten die Hypothese von [Li04]. Selbst unter Einbeziehung historischer Daten lieferte die Theil-Statistik für 39 von 88 Modellrechnungen (das entspricht 44% aller durchgeführten Modellrechnungen) Werte größer oder gleich 1. Damit sind 39 von 88 Modellrechnungen schlechter als die statische Prognose. Gleitender Durchschnitt und exponentielles Glätten schnitten dabei im Experiment etwa gleich schlecht ab.

Abschließend stellte [Li04] daher fest, dass das Weibull-Modell tatsächlich gut geeignet ist, um Fehlerraten bzw. Feldfehlermuster zu modellieren. Naive Parameterextrapolation führt hingegen zu keinen verwertbaren Ergebnissen, da sie unabhängig von Prediktoren funktioniert. [Li04] stützt damit die Hypothese 1 dieser Arbeit, dass gute Feldfehlerprognosen wesentlich von der Wahl (und Verwendung) geeigneter Prediktoren abhängen.

Weiters zeigt [Li04], dass auch ohne den Einsatz von Source-Code-Metriken gute Ergebnisse bei der Modellierung und Vorhersage von Feldfehlerraten erzielt werden können. Voraussetzung dazu ist aber die Auswahl eines den Entwicklungsprozess und andere Rahmenbedingungen möglichst gut abbildenden statistischen Modells. D.h., dass implizit qualitatives Wissen über die Entwicklungsumgebung, den Software-Entwicklungsprozess und anderen Rahmenbedingungen in die Modellauswahl einfließen müssen. Die Verwendung von Modellen mit Parameterauswahl, wie sie z.B. von [Li06] später angewandt wurde, erlaubt einen quantifizierbaren und damit einerseits a priori besser überprüfbaren und andererseits universelleren Ansatz zur Feldfehlerprognose.

3.7 Fehlerverfolgung und Reliability-Modeling

Shanker Sanyal, Kei Aida, Kostas Gaitanos, George Wowk und Sam Lahiri vom IBM Canada Limited Laboratory (North York, Ontario, Canada) berichten in [Sany92] ebenfalls über die Prognose der Anzahl von Feldfehlern in einer neuen Software-Produkt-Release. Hintergrund für die Studien war das Ziel von IBM, qualitativ hochwertige Softwaresysteme herzustellen.

Die Feldfehler wurden von [Sany92] mit CMVC, einem IBM-eigenen Werkzeug für Configuration-Management und Version-Control, erfasst und ausgewertet. Ein Feldfehler konnte dabei einen der folgenden Stati einnehmen:

- Open
- Working (in Bearbeitung)
- Verify (in Qualitätssicherung)
- Closed

Die Feldfehler sind eingeteilt in die Fehlerklassen:

- 1 Kritisch
- 2 Nicht kritisch
- 3 Unerwartetes Programmverhalten
- 4 Empfehlung bzw. Change-Request

Zusätzlich wurden die Feldfehler mit dem Customer-Oriented-Reliability-Estimate (CORE) Schema eingeteilt. Dabei wurde jedem Feldfehler ein Gewicht nach folgender Formel zugeordnet:

Formel 3-7: Fehlergewicht nach CORE [Sany92]

$$\text{Weight} = \text{Frequency} + \text{Severity} + \text{Workaround}$$

Mit Frequency (Wahrscheinlichkeit des Auftretens):

- 0 Keine Auswirkung auf einen Benutzer
- 1 Kann eine Auswirkung auf einen Benutzer haben
- 2 Hat Auswirkung auf wenige Benutzer
- 3 Hat Auswirkung auf einige Benutzer
- 4 Hat Auswirkung auf viele Benutzer
- 5 Hat Auswirkung auf fast alle Benutzer (Kernfunktion betroffen)

Mit Severity (Konsequenzen des Feldfehlers):

- 0 Programmverhalten könnte „schöner“ sein
- 1 Seltenes Auftreten von störendem oder unerwartetem Programmverhalten
- 2 Ständiges Auftreten von störenden oder unerwartetem Programmverhalten
- 3 Falsche Ergebnisse, führen jedoch nicht zu kritischem Programmverhalten
- 4 Falsche Ergebnisse, die zu kritischem Programmverhalten führen

Mit Workaround (wie einfach kann der Fehler vermieden bzw. umgangen werden):

- -2 Workaround nicht nötig bzw. kann der Fehler mit einfachsten Mitteln umgangen werden
- -1 Workaround möglich
- 0 Kein Workaround möglich

Daraus ergibt sich ein minimales CORE-Gewicht von -2 sowie ein maximales CORE-Gewicht von 9.

Feldfehlerdaten wurden von [Sany92] wöchentlich aus dem CMVC-Werkzeug ausgewertet. Zum einen im Rahmen eines Fehlerstatus-Diagramms, das den kumulierten Fehlerstatus einer Software-Komponente je Woche zeigte. Zum anderen im Rahmen eines Fehlerratendiagramms.

Aus dem Fehlerstatus-Diagramm kann man die Entwicklung der offenen (neuen) Feldfehler gegenüber den abgeschlossenen Feldfehlern ablesen. Wird der Abstand immer größer, sind offenbar zu wenige Ressourcen mit der Abarbeitung der Fehlermeldungen beschäftigt.

Das Fehlerratendiagramm zeigt die wöchentliche Fehlerrate einer Software-Komponente, normalisiert auf 1000 Code-Zeilen.

Weiters wurde von [Sany92] die Intensität der Verwendung des betrachteten Software-Produktes in Stunden je Woche aufgezeichnet, um genauere Aussagen über die Entwicklung der Fehlerraten machen zu können: [Sany92] ging wie [Li04] davon aus, dass bei intensiverer Beschäftigung mit einem Software-Produkt auch mehr Fehler entdeckt werden.

Auf Basis der ausgewerteten Feldfehlerdaten versuchte [Sany92] nun ein Reliability-Model festzulegen. Ziel des Reliability-Models sollte es sein, die Anzahl von Authorized Program Analysis Reports (APARs) in einem Software-Produkt abzubilden und auch prognostizieren zu können. Das Modell sollte dabei auch den Entwicklungsprozess berücksichtigen und möglichst gut abbilden können.

Basis für das Modell war ein 5-fach gleitender Durchschnitt der wöchentlichen Feldfehlerrate. Gleitender Durchschnitt deshalb, um ungewünschte Abweichungen bzw. plötzliche Spitzen zu glätten und die Tatsache zu berücksichtigen, dass die Wahrscheinlichkeit, dass ein Fehler zwei Wochen vor oder nach seinem eigentlichen Auftreten gemeldet wird, größer als Null ist.

Da nicht jeder Feldfehler in einen APAR münden muss, wurde von [Sany92] auf Basis von Stichprobenuntersuchungen auch noch der Prozentsatz an Feldfehlern einer bestimmten Fehlerklasse festgelegt, die in einen APAR münden:

- 100% der Feldfehler mit Fehlerklasse 1
- 90% der Feldfehler mit Fehlerklasse 2
- 70% der Feldfehler mit Fehlerklasse 3
- 20% der Feldfehler mit Fehlerklasse 4

[Sany92] wählte das Reliability-Model aus folgenden Modellen:

- Logarithmisches Poisson-Modell

Formel 3-8: Logarithmisches Poisson-Modell [Sany92]

$$u(t) = \theta(1 - e^{\frac{-\lambda t}{\theta}})$$

Mit θ als Gesamtanzahl aller Feldfehler

Mit λ als initialer Feldfehlerrate

- Inflection-Model (Beugungsmodell)

Formel 3-9: Inflection-Model [Sany92]

$$u(t) = \theta \frac{(1 - e^{-\phi t})}{(1 + \psi e^{-\phi t})}$$

Mit ϕ entspricht ungefähr λ/θ

Mit ψ als Beugungsrate (für ψ=0 geht das Beugungsmodell in das exponentielle Modell über)

Dieses Modell eignet sich, wenn Fehler gehäuft auftreten.

- Hyperexponentielles Modell

Formel 3-10: Hyperexponentielles Modell [Sany92]

$$u(t) = \theta_1(1 - e^{\frac{-\lambda_1 t}{\theta_1}}) + \theta_2(1 - e^{\frac{-\lambda_2 t}{\theta_2}})$$

Dieses Modell eignet sich zur Abbildung zweier getrennter Fehlerraten (z.B. wenn neue Code-Teile zu bereits getesteten Codes hinzugefügt werden).

[Sany92] entschied sich für das logarithmische Poisson-Modell, da es die untersuchte Entwicklungsumgebung am besten wiederspiegelte. Da der Anteil an Feldfehlern einer bestimmten Fehlerklasse, die zu einem APAR führen, außerhalb des Modells festgelegt wurde, erfolgte automatisch eine Gleichgewichtung der vier betrachteten Fehlerklassen. Als Zeiteinheit wurden von [Sany92] wöchentliche Intervalle gewählt. Da bereits die Basisdaten über einen 5-fach gleitenden Durchschnitt geglättet wurden, wurde im Modell auch keine Durchschnittsberechnung mehr angewandt.

Die Basisdaten zur Modellerstellung wurden von [Sany92] erst gewonnen, als der zugrundeliegende Source-Code komplett und stabil und die Testabdeckung äquivalent zum angenommenen künftigen Nutzungsgrad war. Um eine Verzerrung der Modellergebnisse durch Doppelzählung oder Fehltypisierung (Change-Requests, die als Fehler gemeldet werden) zu minimieren, wurden alle gemeldeten Feldfehler vor der Annahme genau analysiert.

Ergebnis der Modellrechnung von [Sany92] war eine Tabelle mit wöchentlichen Werten für die Anzahl der in der Software-Produkt-Release verbleibenden Feldfehlern, der Anzahl der neu gemeldeten Feldfehler (Feldfehlerrate) sowie der kumulierten Feldfehleranzahl. Weiters wurden die Gesamtanzahl der Feldfehler sowie die initiale wöchentliche Feldfehlerrate unmittelbar nach Veröffentlichung des Software-Produkts innerhalb eines 75%-Konfidenzintervalls ermittelt.

Die Werthaltigkeit der Ergebnisse wurde von [Sany92] mit statistischen Methoden (Residual-Plot, Wert von R^2) überprüft. Der Wert von R^2 war mit 0,98 nahe bei 1, was einer guten Übereinstimmung des Modells mit der Realität entspricht.

[Sany92] zeigt wie schon [Li04], dass auch ohne den Einsatz von Source-Code-Metriken gute Ergebnisse bei der Modellierung und Vorhersage von Feldfehlerraten erzielt werden können. Aber auch die Untersuchungen von [Sany92] basierten auf umfangreichen historischen Fehlerdaten. Darüberhinaus standen [Sany92] Daten über die Nutzungsintensität der Software-Produkt-Releases zur Verfügung, die als Prediktor für die Modellbildung verwendet wurden. All dies sind Daten, die nachträglich nur schwer oder überhaupt nicht gewonnen werden können. Damit stützt auch [Sany92] zumindest Hypothese 2 dieser Arbeit, dass eine eingeschränkte Verfügbarkeit von Metriken bzw. Prediktoren wegen nachträglicher Datengewinnung zu schlechten Prognosen führt.

Kapitel 4

4 Fallstudie

4.1 Zielsetzung

Untersuchungsgegenstand ist ein großes kommerzielles Datenerfassungs- und Verarbeitungssystem, im Folgenden kurz KDV-System genannt. Das KDV-System ist eine komplette Individualsoftwareentwicklung. Entwickelt wird seit dem Jahr 2003, derzeit laufen gerade die Entwicklungsarbeiten an der Release, die erstmals den kompletten geplanten Funktionsumfang enthalten soll. Anschließend soll der Roll-Out an die Kunden erfolgen.

Das KDV-System ist in 8 große Subsysteme gegliedert und basiert auf einer 3-tier Client-Server-Architektur:

- Client: PowerBuilder (Rich-Client)
- Business-Server: Tuxedo Transaktionsmonitor
- Datenbank-Server: ORACLE

Folgende Entwicklungsumgebungen werden verwendet:

- Power Builder (Client)
- Cobol bzw. MagnaX (Transaktionsmonitor)
- PL/SQL

Die Bausteine des KDV-Systems sind:

- Fenster
- Listbilder
- Batch-Verarbeitungen

Ziel der Fallstudie ist die Modellierung und Vorhersage der Anzahl Feldfehler für künftige Releases eines Subsystems des KDV-Systems. Dabei sollen die in dieser Arbeit aufgestellten Hypothesen anhand eines realen Projektumfelds überprüft

werden. Die Fallstudie basiert dabei maßgeblich auf den Untersuchungen von [Li06], der in einem vergleichbaren Umfeld ebenfalls die Anzahl Feldfehler bestimmen möchte.

4.2 Datengewinnung

Das betrachtete Subsystem ist seit Ende 2007 im produktiven Einsatz. Nach Bereinigung der Daten stehen 30 Releases zur weiteren Untersuchung zur Verfügung (V2.0.04 bis V2.5.11), die in der Zeit zwischen 2005 und 2007 entstanden sind.

Lt. [Li06] sind folgende Metriken von Bedeutung:

- Produktmetriken (vergleiche Tabelle 3-1: Produktmetriken [Li06])
- Entwicklungsmetriken (vergleiche Tabelle 3-2: Entwicklungsmetriken [Li06])
- Verwendungsmetriken (vergleiche Tabelle 3-3: Verwendungsmetriken [Li06])
- Konfigurationsmetriken (Tabelle 3-4: Konfigurationsmetriken [Li06])

Zur Gewinnung der Produkt-Metriken stehen je Release die Source-Code-Dateien

- PowerBuilder
- Cobol
- MagnaX
- PL/SQL

zur Verfügung. Fehlerdaten können aus der Befunddatenbank Merant Tracker (vergleiche auch [Li06]) gewonnen werden.

Die Ermittlung von komplexeren Produkt-Metriken erweist sich als schwierig, da es mir weder für die „exotischen“ Programmiersprachen PowerBuilder und MagnaX, noch für die kommerziellen Programmiersprachen Cobol und PL/SQL gelungen ist, frei erhältliche Werkzeuge zur Gewinnung von Source-Code-Metriken am Markt zu finden. Die Beschaffung kommerzieller Werkzeuge (wie z.B. COBOL FGM von pro et con GmbH etc.) bzw. die Entwicklung eigener Analyseprogramme hätte aber den Rahmen dieser Diplomarbeit gesprengt.

Ich habe mir daher mit dem frei erhältlichen Tool LinesOfCodeWichtel (http://www.andreas-berl.de/linesofcodewichtel/index.html) beholfen. LinesOfCodeWichtel ist ein einfach auf verschiedene Programmiersprachen anzupassendes Werkzeug, dass folgende Größenmetriken ermitteln kann:

- Anzahl der gescannten Dateien
- Anzahl Gesamtzeilen
- Anzahl Codezeilen (Zeilen mit ausführbarem Code, wird im Folgenden auch mit LOC bezeichnet)
- Anzahl Leerzeilen
- Anzahl Kommentarzeilen

Die Anzahl der Dateien reicht von 169 für den Release V2.0.04 bis zu 423 für den Release V2.5.11. Die zugehörige Anzahl an echten Code-Zeilen geht von 78818 bis 263943 LOC.

In der Merant Tracker Befunddatenbank befinden sich zu den betrachteten 30 Releases 2025 Fehlermeldungen. Befundtypen, die nicht dem Typ „Fehlermeldung“ entsprechen, werden nicht berücksichtigt. Die Fehlermeldungen sind in folgende Fehlerklassen eingeteilt:

0 – Kein Befund

1 – Keine direkte Auswirkung

2 – Umgehungslösung vorhanden

3 – Keine Umgehungslösung vorhanden

4 – Kein Betrieb möglich

Die einzelnen Fehlermeldungen können über das Attribut Version eindeutig einer bestimmten Release zugeordnet werden. Leider war es nicht mehr möglich, die genauen Veröffentlichungszeitpunkte der einzelnen Releases nachträglich festzustellen. Eine Unterscheidung, welche Fehler vor der Veröffentlichung und welche danach aufgetreten sind, ist damit nicht möglich. Auch Daten zu den anderen Entwicklungs- metriken nach [Li06] sind nicht verfügbar (z.B. Anzahl von Code-Änderungen, Anzahl und Qualität der beteiligten Personen) und können nachträglich nicht festgestellt werden. Die Anzahl der Codeänderungen kann aber durch Verwendung der Trend-Metrik Δ-LOC (Differenz der Anzahl Code-Zeilen zwischen zwei Releases) angenähert werden.

Aus Sicht der Verwendungsmetriken nach [Li06] wurden nur Minor-Releases betrachtet. Auf Grund der fehlenden Veröffentlichungszeitpunkte können auch die restlichen Verwendungsmetriken nachträglich nicht festgestellt werden.

Die Konfigurationsmetriken nach [Li06] haben für die Fallstudie keine Bedeutung, da das KDV-System nur auf einer bestimmten Plattform zum Einsatz kommt.

4.3 Modellierung

Es soll also die Gesamtanzahl aller Feldfehler einer Release in Abhängigkeit der Produktmetriken (Prediktoren) Anzahl Dateien, Anzahl Code-Zeilen (LOC), Δ-Dateien (Differenz der Anzahl Dateien zwischen zwei Releases) und Δ-LOC (Differenz der Anzahl Code-Zeilen zwischen zwei Releases) modelliert werden. Δ-LOC kann dabei, wie im vorigen Kapitel erwähnt, auch als Näherung der Entwicklungsmetrik (Anzahl der Code-Änderungen) verstanden werden.

Wie bei [Li06] kommt im Rahmen der Fallstudie lineare Regression als Prognosemodell zum Einsatz. Insbesondere bei linearen Regressionsmodellen mit nur einem Prediktor kann das Ergebnis graphisch besonders anschaulich dargestellt werden und trägt damit sehr zum Verständnis der Materie bei.

Als Werkzeug zur statistischen Analyse wurde das frei verfügbare Programm „R-Project“ (http://www.r-project.org) verwendet. Hilfreich bei der Modellierung und Auswertung der Daten war auch das Buch „Statistik aktiv mit R“ von Hans Peter Wolf, Peter Naeve und Veith Tiemann [Wolf06].

4.4 Trainings- und Testdatensätze

Tabelle 4-1 enthält die 26 Trainings- (Release V2.0.05 bis V2.5.07) sowie die 4 Testdatensätze (Release V2.5.08 bis V2.5.11). Je Release sind die Gesamtanzahl der Feldfehler, die Anzahl der Fehler ohne Priorität / mit Priorität 0 / mit Priorität 1 / mit Priorität 2 / mit Priorität 3 / mit Priorität 4 sowie die Metriken Anzahl Dateien, Anzahl Code-Zeilen (LOC) und die Trendmetriken Δ-Dateien und Δ-LOC angegeben.

Tabelle 4-1: Trainings- und Testdatensätze der Fallstudie

Release	# Fehler	<<none>>	0	1	2	3	4	Anzahl Dateie	Codezeilen	Δ Dateien	Δ Codezeilen
V2.0.04	172	33	1	37	31	68	2	169	78818		
V2.0.05	83	12	2	11	4	32	22	188	99228	19	20410
V2.0.06	126	18	7	20	22	34	25	212	110553	24	11325
V2.0.07	104	23	1	16	17	32	15	231	135344	19	24791
V2.0.08	224	5	4	39	29	86	61	278	155888	47	20544
V2.0.12	57	1	0	5	9	20	22	295	166854	17	10966
V2.0.13	66	5	0	0	8	47	6	297	171744	2	4890
V2.0.14	1	0	0	0	1	0	0	297	172552	0	808
V2.1.00	71	5	1	6	11	28	20	310	180324	13	7772
V2.1.01	6	0	0	0	6	0	0	310	180676	0	352
V2.1.02	11	1	0	0	2	5	3	310	181092	0	416
V2.1.03	13	0	0	0	2	11	0	310	181334	0	242
V2.1.04	6	1	0	1	3	1	0	310	180622	0	-712
V2.2.00	216	1	1	35	34	98	47	370	214976	60	34354
V2.2.01	95	0	4	11	15	34	31	370	215589	0	613
V2.3.00	49	4	1	3	9	25	7	390	231569	20	15980
V2.3.01	139	8	3	17	29	58	24	390	233116	0	1547
V2.4.00	14	0	0	3	3	8	0	395	237755	5	4639
V2.5.00	110	4	2	17	20	48	19	395	238011	0	256
V2.5.01	87	2	0	3	13	46	23	397	244865	2	6854
V2.5.02	23	1	0	3	7	9	3	396	254692	-1	9827
V2.5.03	9	0	0	0	4	3	2	396	247307	0	-7385
V2.5.04	23	1	0	2	7	9	4	397	249094	1	1787
V2.5.05	45	1	0	1	12	28	3	397	249698	0	604
V2.5.06	25	0	0	1	3	17	4	398	250472	1	774
V2.5.07	54	2	0	3	8	37	4	408	251810	10	1338
V2.5.08	65	0	0	12	26	20	7	408	253072	0	1262
V2.5.09	36	2	0	4	14	15	1	421	259769	13	6697
V2.5.10	59	4	0	1	10	39	5	422	261180	1	1411
V2.5.11	36	5	0	2	8	21	0	423	263943	1	2763

4.5 Lineare Regression mit allen Prediktoren

Damit festgestellt werden kann, welche Prediktoren besonders signifikant sind, erfolgt zunächst eine lineare Regression mit allen verfügbaren Prediktoren. Die Signifikanz einzelner Prediktoren kann dann von den „Analysis of Variance“ (ANOVA) Tabellen, die „R“ zur Verfügung stellt, direkt abgelesen werden (vergleiche „Signif. Codes“).

Mithilfe eines QQ-Plots, bei dem die empirischen Quantile der geschätzten Residuen gegen die Quantile der Standardnormalverteilung aufgetragen werden, wird auch jeweils die Annahme der Normalverteilung der Residuen überprüft. Die Normalverteilung der Residuen ist eine wesentliche Voraussetzung zum Einsatz der linearen Regression als Modellierungswerkzeug.

Zur Bestimmung der allgemeinen Güte des Modells wird das Bestimmtheitsmaß R^2 herangezogen (vergleiche auch „Multiple R-Squared“). R^2 entspricht einem Wert zwischen 0 und 1. 0 bedeutet keine Übereinstimmung des Modells mit der Realität, während 1 volle Übereinstimmung des Modells mit der Realität bedeutet. Je näher R^2 daher bei 1 liegt, desto besser ist die Güte des Modells.

4.5.1 Lineare Regression der Gesamtanzahl aller Feldfehler

Die Modellparameter ergeben sich zu:

```
lm(formula = fehler.gesamt ~ dateien + loc + dateien.delta +
    loc.delta)

Residuals:
   Min     1Q Median     3Q    Max
-68.09 -25.01  -5.72  21.67  87.70

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -58.653498  81.103116  -0.723   0.4779
dateien         1.434607   1.263250   1.136   0.2695
loc            -0.001936   0.001786  -1.084   0.2914
dateien.delta   2.364390   1.255917   1.883   0.0744 .
loc.delta       0.001074   0.001954   0.550   0.5886
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 40.37 on 20 degrees of freedom
Multiple R-Squared: 0.6414,     Adjusted R-squared: 0.5697
F-statistic: 8.945 on 4 and 20 DF,  p-value: 0.0002604
```

Die Güte des Modells ist mit einem Bestimmtheitsmaß von 0,6414 als mittelmäßig einzustufen. Dieser Wert lässt bereits eine eingeschränkte Verlässlichkeit des Modells vermuten.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.gesamt
              Df Sum Sq Mean Sq F value    Pr(>F)
dateien        1   3105    3105  1.9051  0.182742
loc            1  15531   15531  9.5289  0.005815 **
dateien.delta  1  39188   39188 24.0440 8.584e-05 ***
loc.delta      1    493     493  0.3023  0.588557
Residuals     20  32597    1630
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt ein starke Signifikanz der Prediktoren LOC sowie Δ-Anzahl-Dateien.

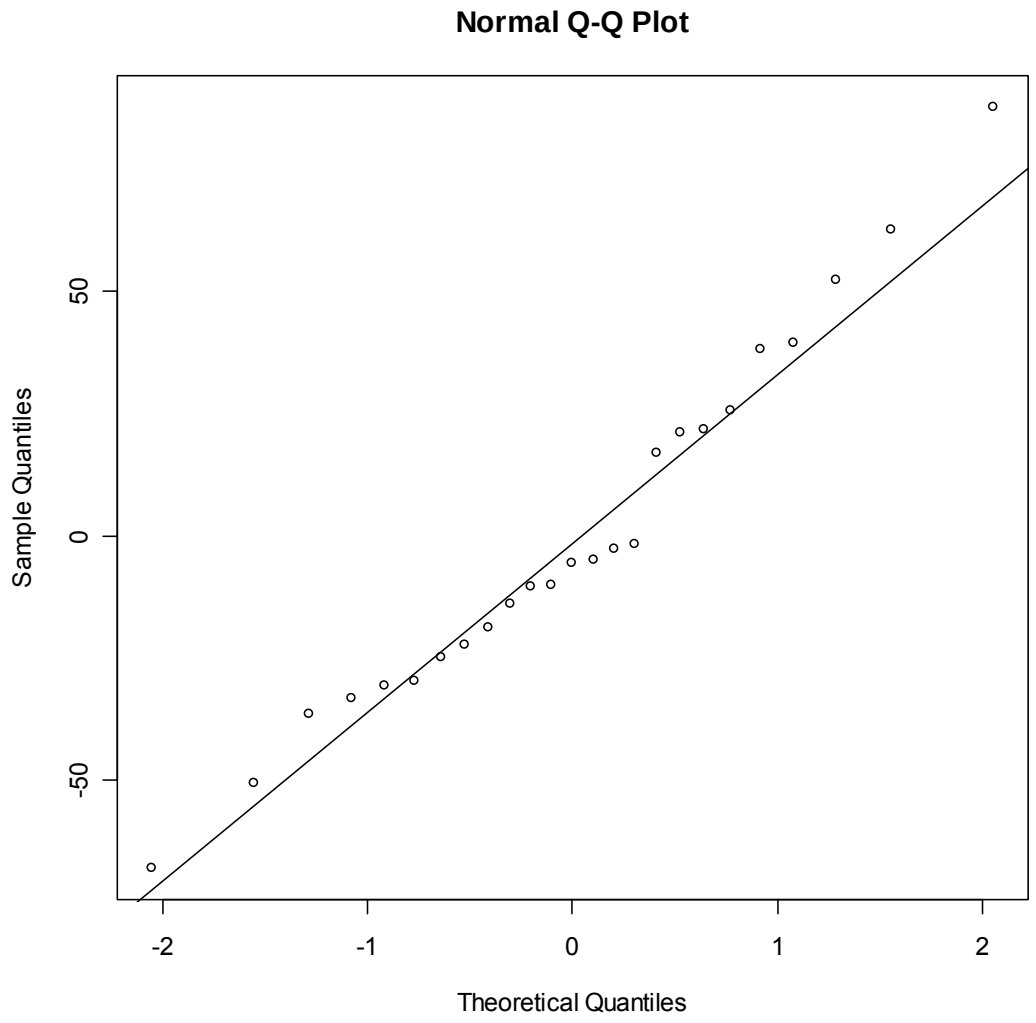

Abbildung 4-1: QQ-Plot Anzahl aller Feldfehler mit allen Prediktoren

Der QQ-Plot in Abbildung 4-1 zeigt, dass die Normalverteilungsannahme der Residuen als Bedingung für die Verwendung der linearen Regression sehr gut erfüllt ist (alle Punkte liegen nahe an der Geraden).

4.5.2 Lineare Regression der Anzahl Feldfehler ohne Priorität

Die Modellparameter ergeben sich zu:

```
lm(formula = fehler.prionone ~ dateien + loc + dateien.delta +
    loc.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-4.6811 -2.8395 -0.3272  1.5997  8.8731

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   23.4465772  8.6587534   2.708   0.0135 *
dateien       -0.1504906  0.1348674  -1.116   0.2777
loc            0.0001484  0.0001907   0.778   0.4457
dateien.delta -0.1168257  0.1340846  -0.871   0.3939
loc.delta      0.0003519  0.0002086   1.687   0.1071
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.31 on 20 degrees of freedom
Multiple R-Squared: 0.548,      Adjusted R-squared: 0.4576
F-statistic: 6.062 on 4 and 20 DF,  p-value: 0.002307
```

Die Güte des Modells ist mit einem Bestimmtheitsmaß von 0,548 gerade noch als mittelmäßig einzustufen und lässt eine eingeschränkte Verlässlichkeit des Modells vermuten.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prionone
              Df Sum Sq Mean Sq F value    Pr(>F)
dateien        1 345.72  345.72 18.6101 0.0003377 ***
loc            1  22.45   22.45  1.2085 0.2846875
dateien.delta  1  29.41   29.41  1.5831 0.2228055
loc.delta      1  52.87   52.87  2.8459 0.1071415
Residuals     20 371.55   18.58
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt eine starke Signifikanz des Prediktors Anzahl-Dateien.

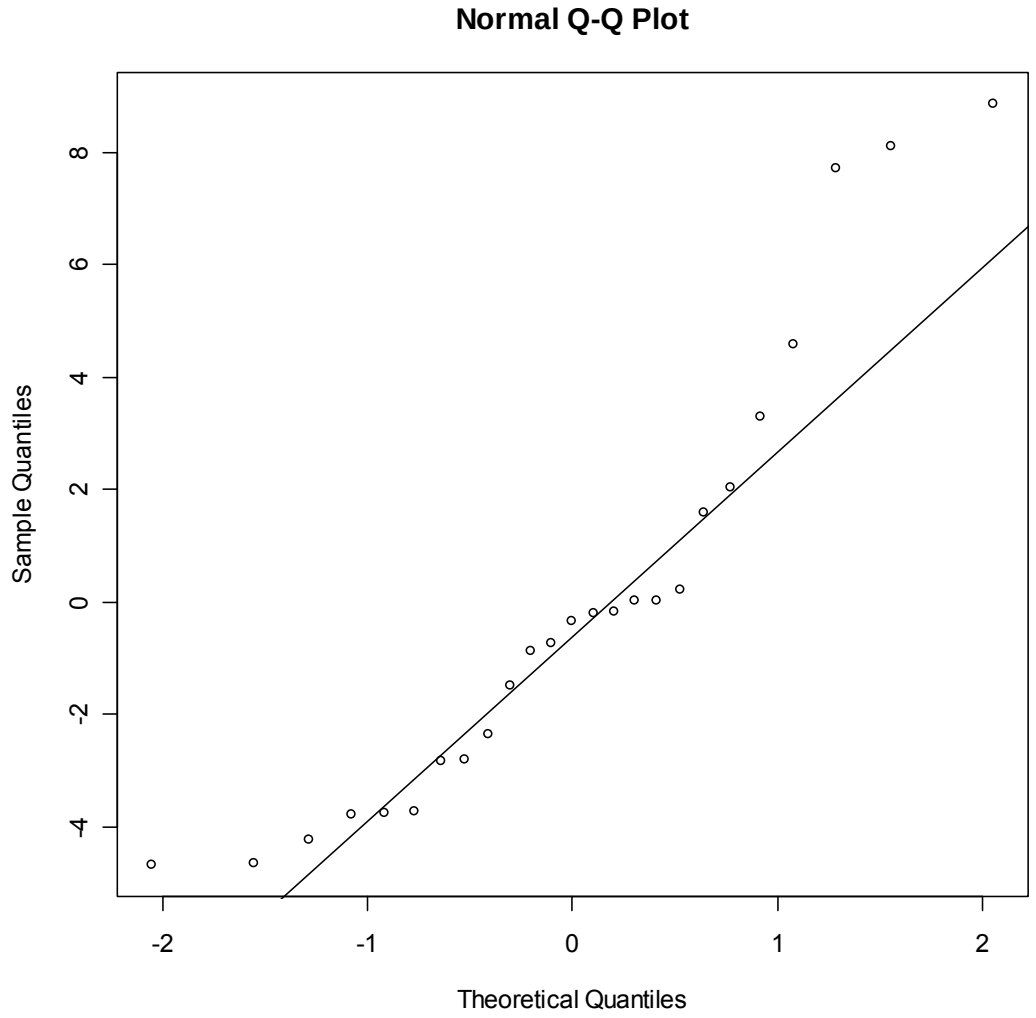

Abbildung 4-2: QQ-plot Anzahl Feldfehler ohne Priorität mit allen Prediktoren

Der QQ-Plot in Abbildung 4-2 zeigt, dass die Normalverteilungsannahme der Residuen nicht mehr optimal erfüllt ist. Die Ergebnisse des aus der linearen Regression resultierenden Modells sind in diesem Fall mit Vorsicht zu behandeln.

4.5.3 Lineare Regression der Anzahl Feldfehler mit Priorität 0

Die Modellparameter ergeben sich zu:

```
lm(formula = fehler.prio0 ~ dateien + loc + dateien.delta + loc.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9805 -1.2458 -0.1121  0.2389  4.0592

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.191e+00  3.192e+00   0.373    0.713
dateien        4.941e-02  4.972e-02   0.994    0.332
loc           -8.415e-05  7.031e-05  -1.197    0.245
dateien.delta  4.886e-02  4.944e-02   0.988    0.335
loc.delta     -5.242e-05  7.691e-05  -0.682    0.503

Residual standard error: 1.589 on 20 degrees of freedom
Multiple R-Squared: 0.3262,     Adjusted R-squared: 0.1915
F-statistic: 2.421 on 4 and 20 DF,  p-value: 0.08216
```

In diesem Fall zeigt das Bestimmtheitsmaß mit einem Wert von 0,3262, dass dieses Modell nicht verwendbar ist.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio0
              Df Sum Sq Mean Sq F value  Pr(>F)
dateien        1 11.244  11.244  4.4527 0.04764 *
loc            1 10.376  10.376  4.1087 0.05620 .
dateien.delta  1  1.663   1.663  0.6585 0.42665
loc.delta      1  1.173   1.173  0.4646 0.50332
Residuals     20 50.505   2.525
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Wie man in diesem Fall sieht, zeigt die ANOVA auch nur eine schwache Signifikanz der Prediktoren Anzahl-Dateien und LOC.

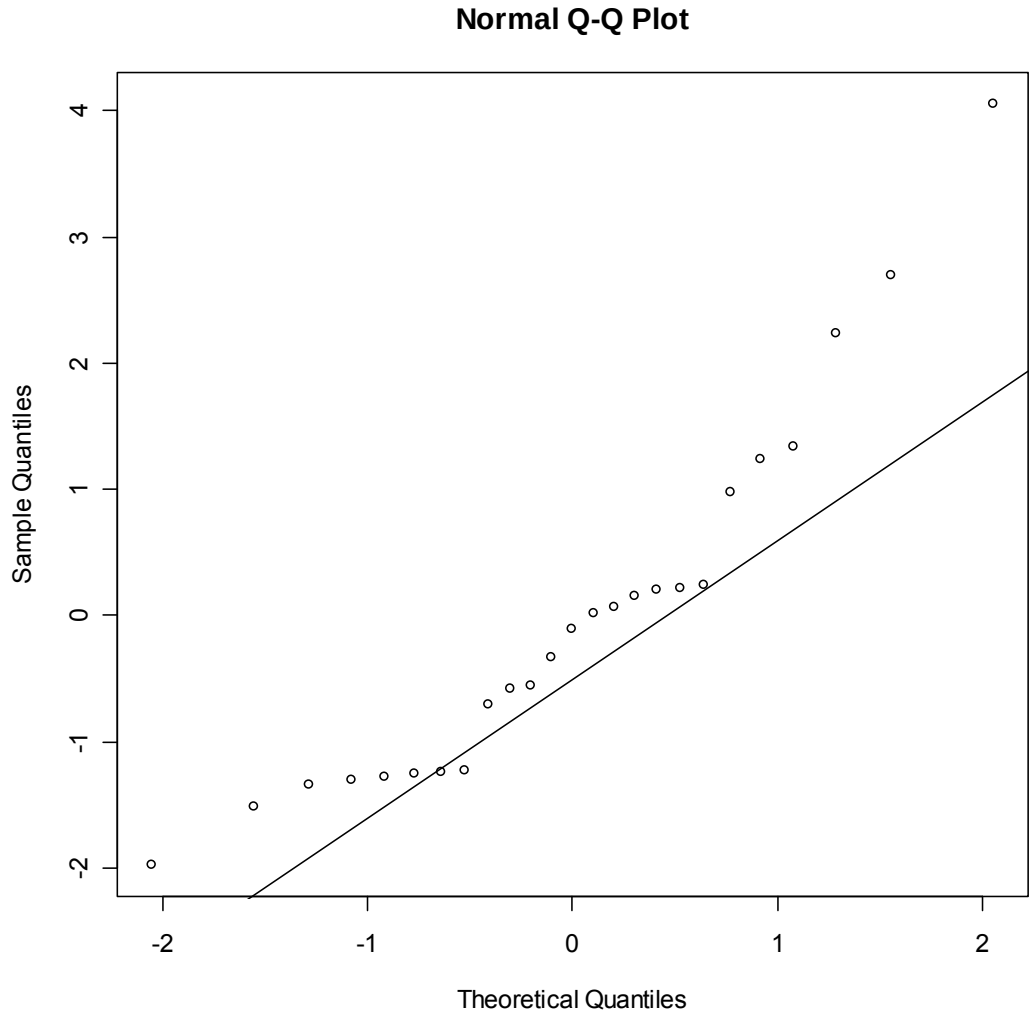

Abbildung 4-3: QQ-Plot Anzahl Feldfehler mit Priorität 0 mit allen Prediktoren

Der QQ-Plot in Abbildung 4-3 zeigt, dass außerdem die Normalverteilungsannahme der Residuen nicht optimal erfüllt ist.

4.5.4 Lineare Regression der Anzahl Feldfehler mit Priorität 1

Die Modellparameter ergeben sich zu:

```
lm(formula = fehler.prio1 ~ dateien + loc + dateien.delta + loc.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-11.668  -2.867  -1.912   3.356  13.648

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3.900e+00  1.324e+01  -0.294   0.7714
dateien       1.248e-01  2.063e-01   0.605   0.5519
loc          -1.767e-04  2.917e-04  -0.606   0.5516
dateien.delta 5.663e-01  2.051e-01   2.761   0.0120 *
loc.delta    -3.339e-05  3.191e-04  -0.105   0.9177
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.593 on 20 degrees of freedom
Multiple R-Squared: 0.6876,     Adjusted R-squared: 0.6251
F-statistic: 11.01 on 4 and 20 DF,  p-value: 6.97e-05
```

Die Güte dieses Modells ist mit einem Bestimmtheitsmaß von 0,6876 als mittelmäßig einzustufen. Der Wert ist zwar besser als beim Modell der Gesamtanzahl aller Feldfehler, lässt aber trotzdem eine eingeschränkte Verlässlichkeit des Modells vermuten.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio1
              Df  Sum Sq Mean Sq F value    Pr(>F)
dateien        1  169.31  169.31  3.8955  0.062391 .
loc            1  411.06  411.06  9.4577  0.005969 **
dateien.delta  1 1332.53 1332.53 30.6591 2.026e-05 ***
loc.delta      1    0.48    0.48  0.0110  0.917698
Residuals     20  869.26   43.46
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt eine starke Signifikanz der Prediktoren Δ-Anzahl-Dateien und LOC. Der Prediktor Anzahl-Dateien ist hier nur schwach signifikant.

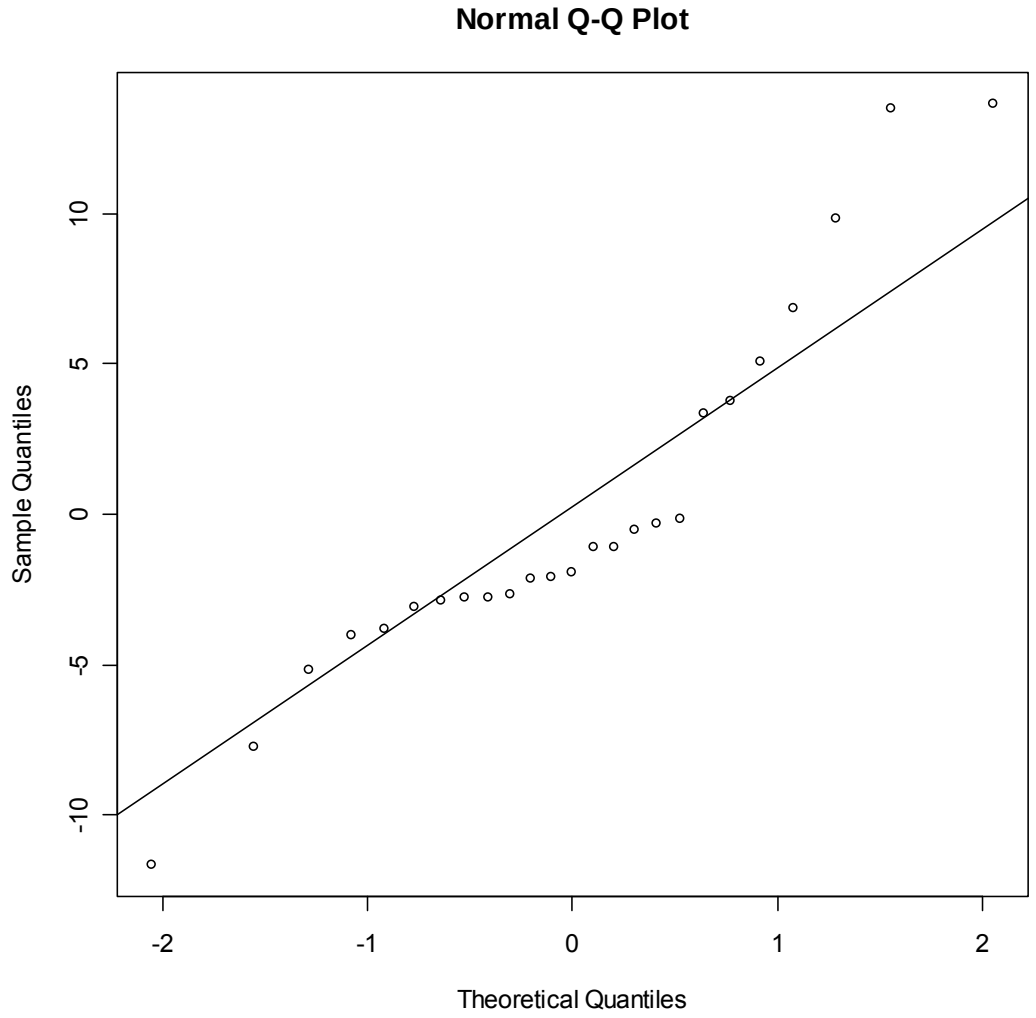

Abbildung 4-4: QQ-Plot Anzahl Feldfehler mit Priorität 1 mit allen Prediktoren

Der QQ-Plot in Abbildung 4-4 zeigt wieder eine bessere Übereinstimmung mit der Normalverteilungsannahme der Residuen. Die Ergebnisse des aus der linearen Regression resultierenden Modells sind ebenfalls mit Vorsicht zu behandeln.

4.5.5 Lineare Regression der Anzahl Feldfehler mit Priorität 2

Die Modellparameter ergeben sich zu:

```
lm(formula = fehler.prio2 ~ dateien + loc + dateien.delta + loc.delta)

Residuals:
     Min       1Q   Median       3Q      Max
-10.5994  -4.4335  -0.9327   3.9048  18.3158

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.589e+01  1.436e+01  -1.107    0.282
dateien        2.724e-01  2.237e-01   1.218    0.237
loc           -3.428e-04  3.163e-04  -1.084    0.291
dateien.delta  3.039e-01  2.224e-01   1.367    0.187
loc.delta      1.598e-04  3.459e-04   0.462    0.649

Residual standard error: 7.148 on 20 degrees of freedom
Multiple R-Squared: 0.5065,     Adjusted R-squared: 0.4078
F-statistic: 5.131 on 4 and 20 DF,  p-value: 0.005198
```

In diesem Fall ist die Güte des Modells mit einem Bestimmtheitsmaß von 0,5065 gerade noch als mittelmäßig einzustufen.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio2
              Df  Sum Sq Mean Sq F value   Pr(>F)
dateien        1    0.41    0.41  0.0080 0.929527
loc            1  347.74  347.74  6.8058 0.016804 *
dateien.delta  1  689.68  689.68 13.4980 0.001506 **
loc.delta      1   10.90   10.90  0.2133 0.649177
Residuals     20 1021.91   51.10
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt eine starke bis mittlere Signifikanz der Prediktoren Δ-Anzahl-Dateien und LOC.

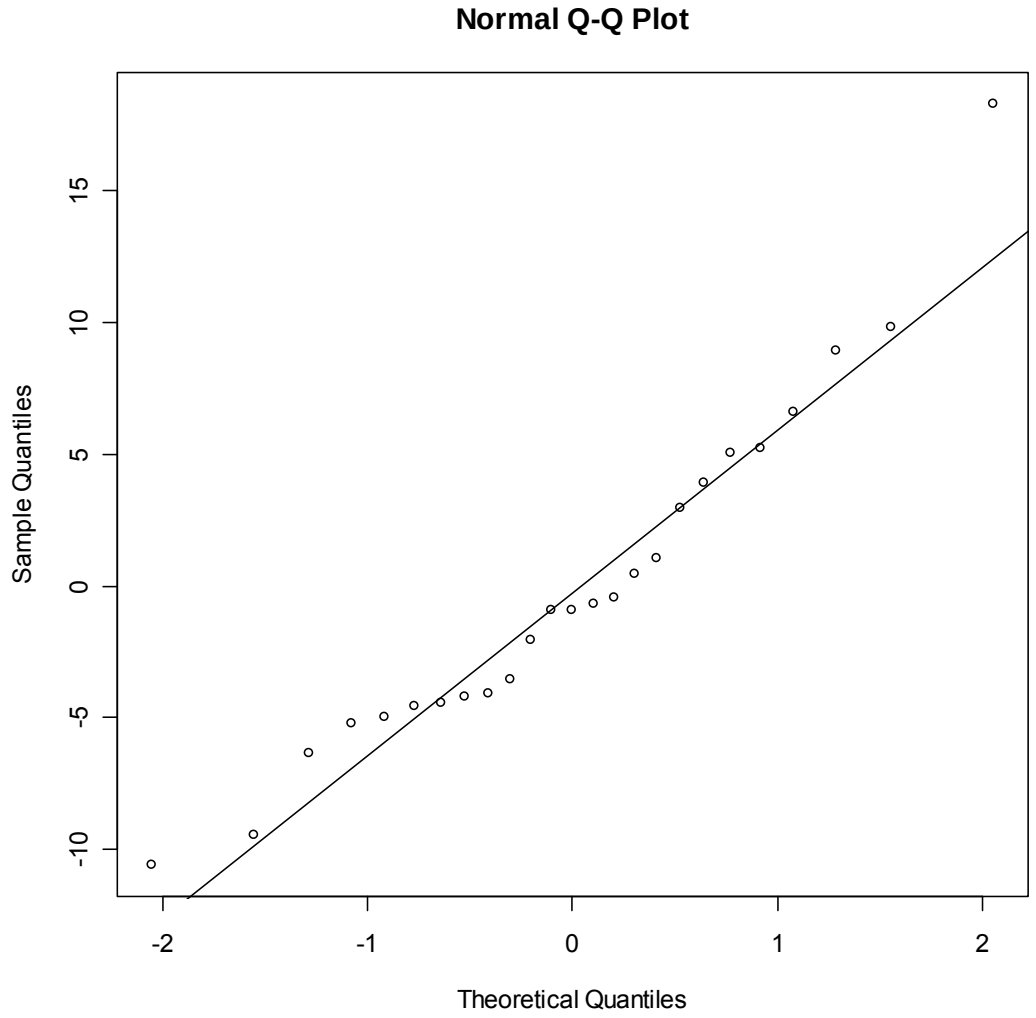

Abbildung 4-5: QQ-Plot Anzahl Feldfehler mit Priorität 2 mit allen Prediktoren

Der QQ-Plot in Abbildung 4-5 zeigt eine fast ideale Übereinstimmung mit der Normalverteilungsannahme der Residuen. Wegen des gerade noch mittelmäßigen Bestimmtheitsmaßes sind die Ergebnisse dieses Modells trotzdem mit Vorsicht zu behandeln.

4.5.6 Lineare Regression der Anzahl Feldfehler mit Priorität 3

Die Modellparameter ergeben sich zu:

```
lm(formula = fehler.prio3 ~ dateien + loc + dateien.delta + loc.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-29.169  -9.758  -2.656   7.037  31.543

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -4.902e+01  3.482e+01  -1.408    0.175
dateien        7.172e-01  5.424e-01   1.322    0.201
loc           -8.801e-04  7.669e-04  -1.148    0.265
dateien.delta  8.722e-01  5.392e-01   1.618    0.121
loc.delta      6.170e-04  8.389e-04   0.736    0.471

Residual standard error: 17.33 on 20 degrees of freedom
Multiple R-Squared: 0.6118,     Adjusted R-squared: 0.5342
F-statistic: 7.881 on 4 and 20 DF,  p-value: 0.0005527
```

Das Bestimmtheitsmaß liegt mit einem Wert von 0,6118 zwar im Bereich der besten bisher erzielten Ergebnisse, eine eingeschränkte Verlässlichkeit lässt sich aber auch hier vermuten.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio3
              Df Sum Sq Mean Sq F value    Pr(>F)
dateien        1    4.7     4.7  0.0157 0.9015588
loc            1 2691.6  2691.6  8.9591 0.0071854 **
dateien.delta  1 6612.3  6612.3 22.0097 0.0001402 ***
loc.delta      1  162.6   162.6  0.5411 0.4705267
Residuals     20 6008.6   300.4
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt eine starke Signifikanz der Prediktoren Δ-Anzahl-Dateien und LOC.

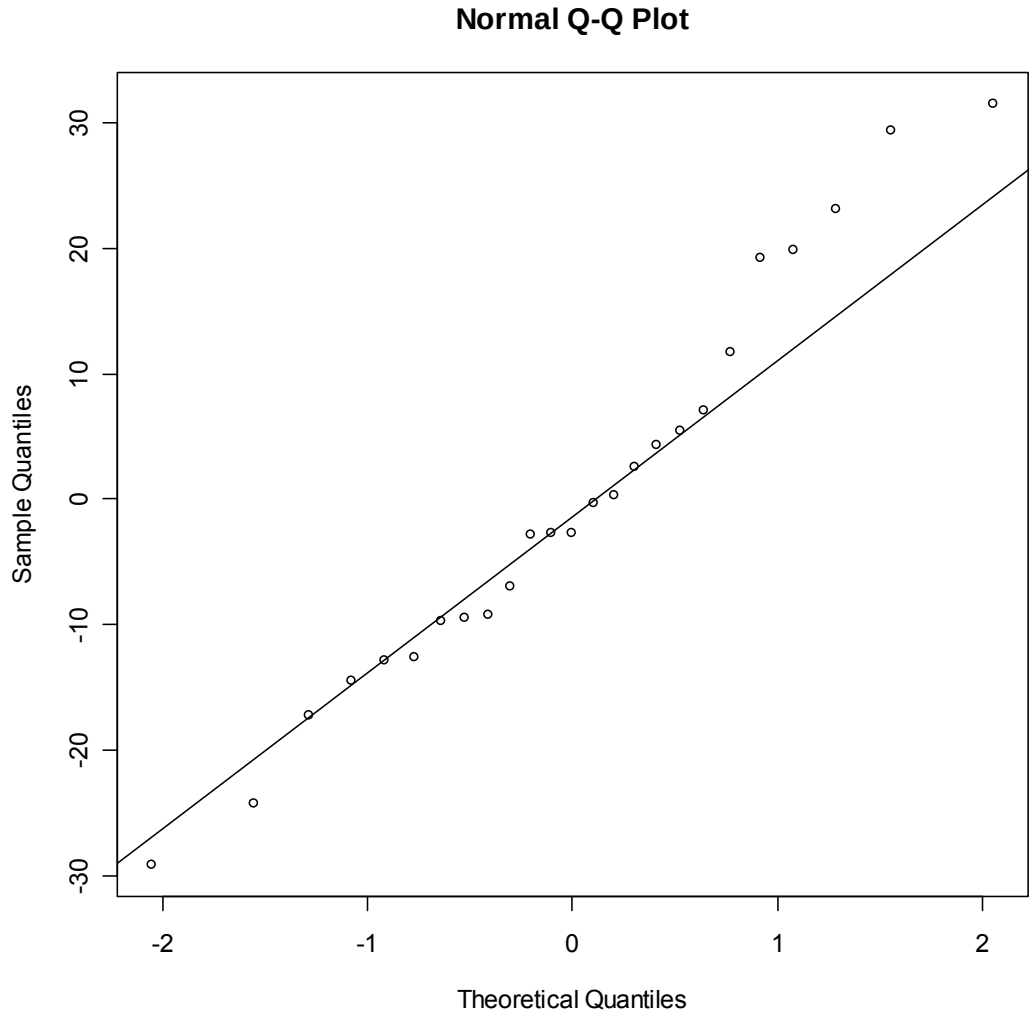

Abbildung 4-6: QQ-Plot Anzahl Feldfehler mit Priorität 3 mit allen Prediktoren

Auch der QQ-Plot in Abbildung 4-6 zeigt eine gute Übereinstimmung mit der Normalverteilungsannahme der Residuen.

4.5.7 Lineare Regression der Anzahl Feldfehler mit Priorität 4

Die Modellparameter ergeben sich zu:

```
lm(formula = fehler.prio4 ~ dateien + loc + dateien.delta + loc.delta)

Residuals:
     Min       1Q   Median       3Q      Max
-18.1402  -7.2976  -0.3942   3.0847  18.9724

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.448e+01  2.098e+01  -0.690   0.4982
dateien        4.213e-01  3.268e-01   1.289   0.2122
loc           -6.001e-04  4.622e-04  -1.298   0.2089
dateien.delta  6.898e-01  3.249e-01   2.123   0.0464 *
loc.delta      3.127e-05  5.055e-04   0.062   0.9513
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.45 on 20 degrees of freedom
Multiple R-Squared: 0.635,      Adjusted R-squared: 0.562
F-statistic: 8.698 on 4 and 20 DF,  p-value: 0.0003087
```

Die Güte des Modells liegt mit einem Bestimmtheitsmaß von 0,635 im Bereich der besten bisher erzielten Ergebnisse.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio4
              Df  Sum Sq Mean Sq F value    Pr(>F)
dateien        1  322.29  322.29  2.9539 0.1011147
loc            1 1248.63 1248.63 11.4441 0.0029555 **
dateien.delta  1 2224.52 2224.52 20.3884 0.0002111 ***
loc.delta      1    0.42    0.42  0.0038 0.9512838
Residuals     20 2182.14  109.11
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt eine starke Signifikanz der Prediktoren Δ-Anzahl-Dateien und LOC.

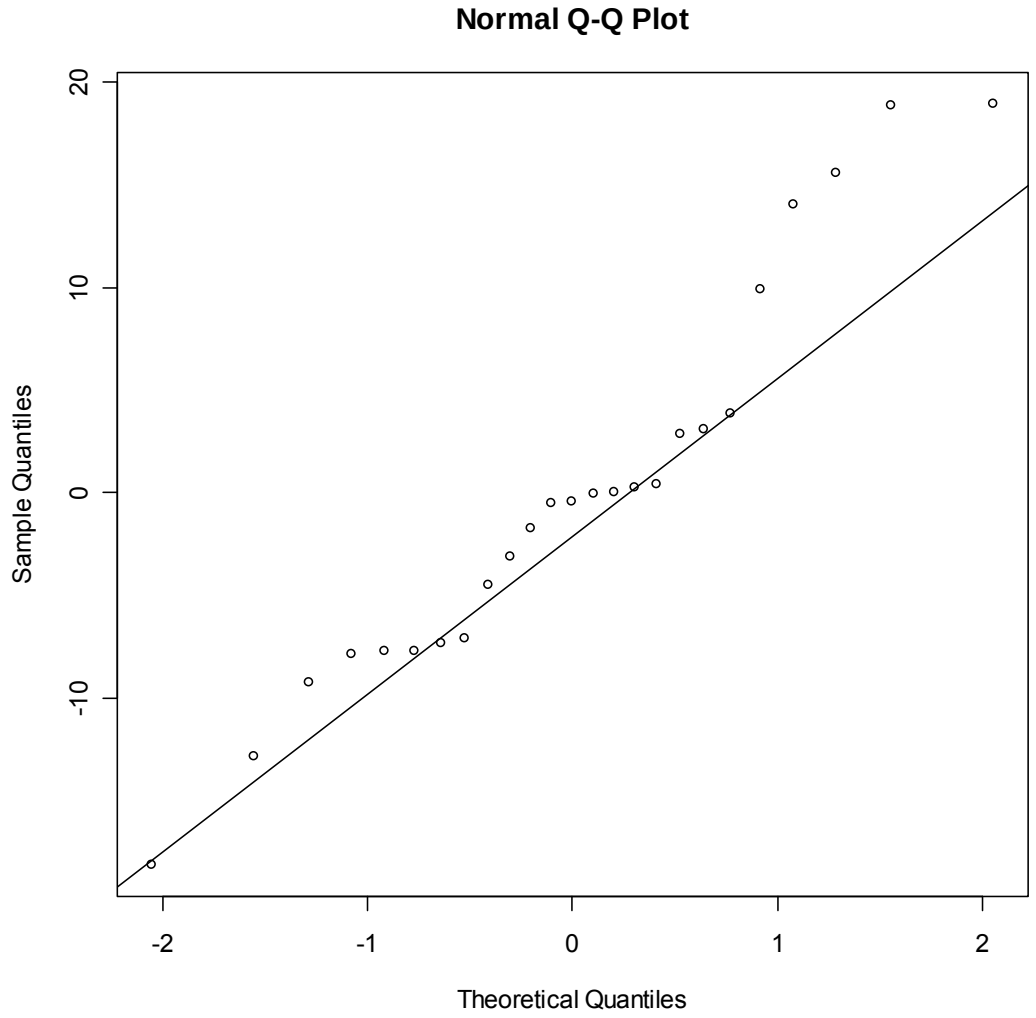

Abbildung 4-7: QQ-Plot Anzahl Feldfehler mit Priorität 4 mit allen Prediktoren

Der QQ-Plot in Abbildung 4-7 zeigt eine weniger gute Übereinstimmung mit der Normalverteilungsannahme der Residuen. Die Ergebnisse des aus der linearen Regression resultierenden Modells für den wichtigen Bereich der Feldfehler mit Priorität 4 sind daher auch in Zusammenhang mit dem bestenfalls mittelmäßigen Bestimmt-heitsmaß mit Vorsicht zu interpretieren.

4.5.8 Schlussfolgerungen

Bei allen bisher durchgeführten linearen Regressionen, ausgenommen Anzahl Feldfehler ohne Priorität und mit Priorität 0, haben sich die Prediktoren LOC sowie Δ-Anzahl-Dateien als besonders signifikant erwiesen.

Tabelle 4-2: Bestimmtheitsmaße Lineare Regression mit allen Prediktoren

	Alle Fehler	**Ohne Prio**	**Prio 0**	**Prio 1**	**Prio 2**	**Prio 3**	**Prio 4**
R^2	0,6414	0,548	0,3262	0,6876	0,5065	0,6118	0,635

Tabelle 4-2 zeigt die zugehörigen Bestimmtheitsmaße noch einmal im Überblick. Das beste Bestimmtheitsmaß gehört dem Modell der Feldfehler mit Priorität 1, dicht gefolgt vom Modell der Gesamtanzahl aller Feldfehler. Das schlechteste Bestimmtheitsmaß hat das Modell der Feldfehler mit Priorität 0. Da die Prognose von Feldfehlern ohne Priorität sowie mit Priorität 0 für die Feldfehlerprognose keine Relevanz besitzt (es handelt sich hier in der Regel um Befunde, die nicht mit dem Auftreten eines Fehlers zusammenhängen) und darüber hinaus die Normalverteilungsannahme der Residuen für diese beiden Fehlerklassen nicht optimal gegeben ist, werde ich mich in weiterer Folge auf die Betrachtung der Feldfehlerprognose der Gesamtfehleranzahl sowie der Anzahl Feldfehler mit Priorität 1 bis 4 beschränken.

Dabei lege ich zunächst mein Hauptaugenmerk auf die Prediktoren LOC sowie Δ-Anzahl-Dateien, wenngleich die Vermutung naheliegt, dass auch Δ-LOC in erster Näherung als Änderungsmetrik betrachtet werden kann und daher nach [Knab06] ein geeigneter Prediktor sein müsste.

4.6 Lineare Regression mit ausgesuchten Prediktoren

4.6.1 Detailanalyse der Gesamtanzahl aller Feldfehler

Die Modellparameter ergeben sich zu:

```
lm(formula = fehler.gesamt ~ loc + dateien.delta)

Residuals:
   Min     1Q Median     3Q    Max
-52.78 -28.29 -13.71  20.64 100.74

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.063e+01  4.077e+01   0.506    0.618
loc           7.561e-05  1.875e-04   0.403    0.691
dateien.delta 3.182e+00  5.580e-01   5.702  9.8e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 39.73 on 22 degrees of freedom
Multiple R-Squared: 0.6181,     Adjusted R-squared: 0.5834
F-statistic:  17.8 on 2 and 22 DF,  p-value: 2.520e-05
```

Das Bestimmtheitsmaß von 0,6181 ist verglichen mit dem Modell auf Basis aller Prediktoren ein wenig schlechter, aber immer noch im Bereich der bisher besten Ergebnisse. Es ist aber auch nicht deutlich besser und lässt daher auch im Fall der Modellierung mit ausgesuchten Prediktoren eine eingeschränkte Verlässlichkeit vermuten.

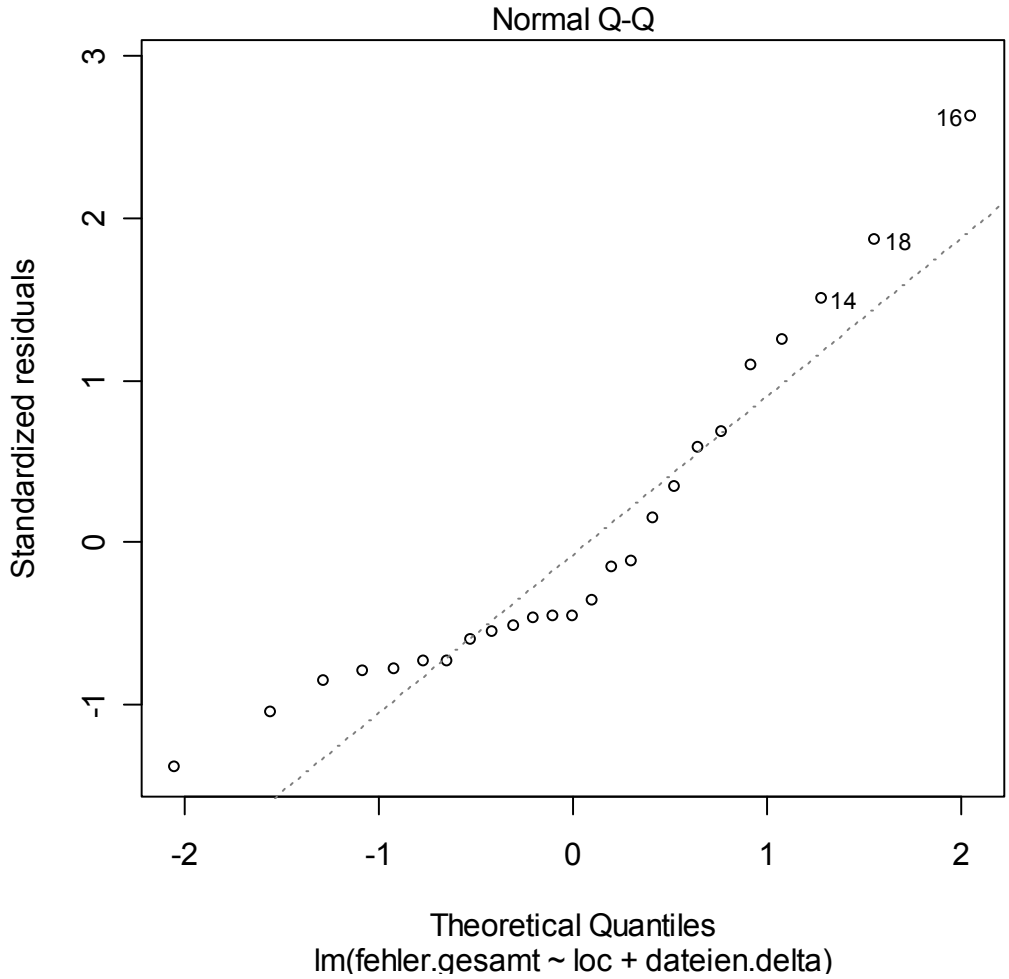

Abbildung 4-8: QQ-Plot Gesamtanzahl aller Feldfehler mit 2 Prediktoren

Der QQ-Plot in Abbildung 4-8 zeigt, dass die für die lineare Regression notwendige Normalverteilungsannahme der Residuen annähernd erfüllt ist. Ein Vergleich mit dem QQ-Plot des Modells Gesamtanzahl aller Feldfehler mit allen Prediktoren (Abbildung 4-1) zeigt aber, dass dort eine bessere Übereinstimmung mit der Normalverteilungsannahme gegeben ist.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.gesamt
              Df Sum Sq Mean Sq F value    Pr(>F)
loc            1   4874    4874  3.0882   0.09277 .
dateien.delta  1  51320   51320 32.5184 9.798e-06 ***
Residuals     22  34720    1578
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die Signifikanz von Δ-Anzahl-Dateien ist hier noch immer sehr hoch, wahrend die Signifikanz von LOC stark abgenommen hat. Zur einfacheren graphischen Darstellung bzw. Analyse betrachten wir im Folgenden daher zunächst die lineare Regression mit der erklärenden Variablen Δ-Anzahl-Dateien.

Lineare Regression mit Δ-Anzahl-Dateien:

```
lm(formula = fehler.gesamt ~ dateien.delta)

Residuals:
   Min     1Q Median     3Q    Max
-49.66 -27.63 -12.56  14.94 102.37

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    36.6325     9.2046   3.980 0.000591 ***
dateien.delta   3.1012     0.5113   6.065 3.47e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 39 on 23 degrees of freedom
Multiple R-Squared: 0.6153,     Adjusted R-squared: 0.5985
F-statistic: 36.78 on 1 and 23 DF,  p-value: 3.473e-06
```

Die Güte des Modells auf Basis Δ-Anzahl-Dateien ist, wie am Bestimmtheitsmaß von 0,6153 ersichtlich, in etwa mit der des Modells auf Basis zweier Prediktoren vergleichbar.

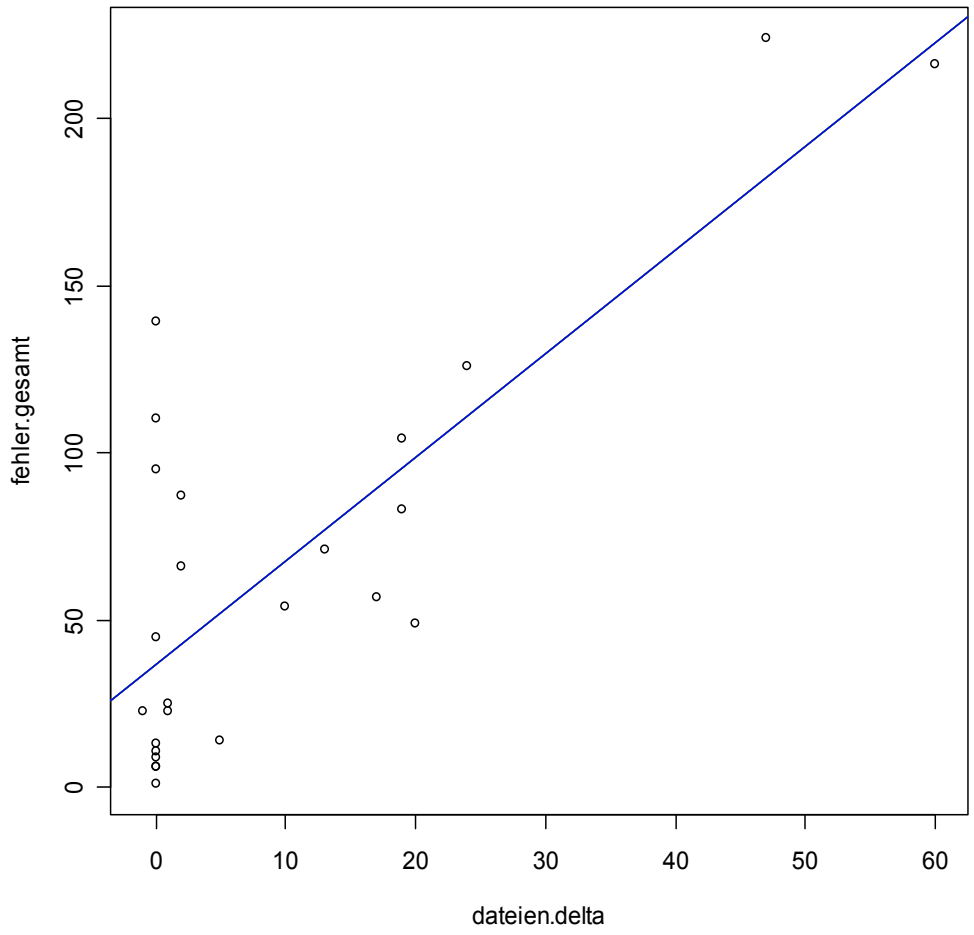

Abbildung 4-9: DOT-Plot Gesamtanzahl aller Feldfehler mit Δ-Anzahl-Dateien

Im DOT-Plot der Abbildung 4-9 liegt der Achsenabschnitt der Regressionsgeraden bei 36,633 und ergibt sich, wie auch aus dem Diagramm ersichtlich, aus einer relativ breiten Streuung der Gesamtfehleranzahl bei Δ-Anzahl-Dateien = 0.

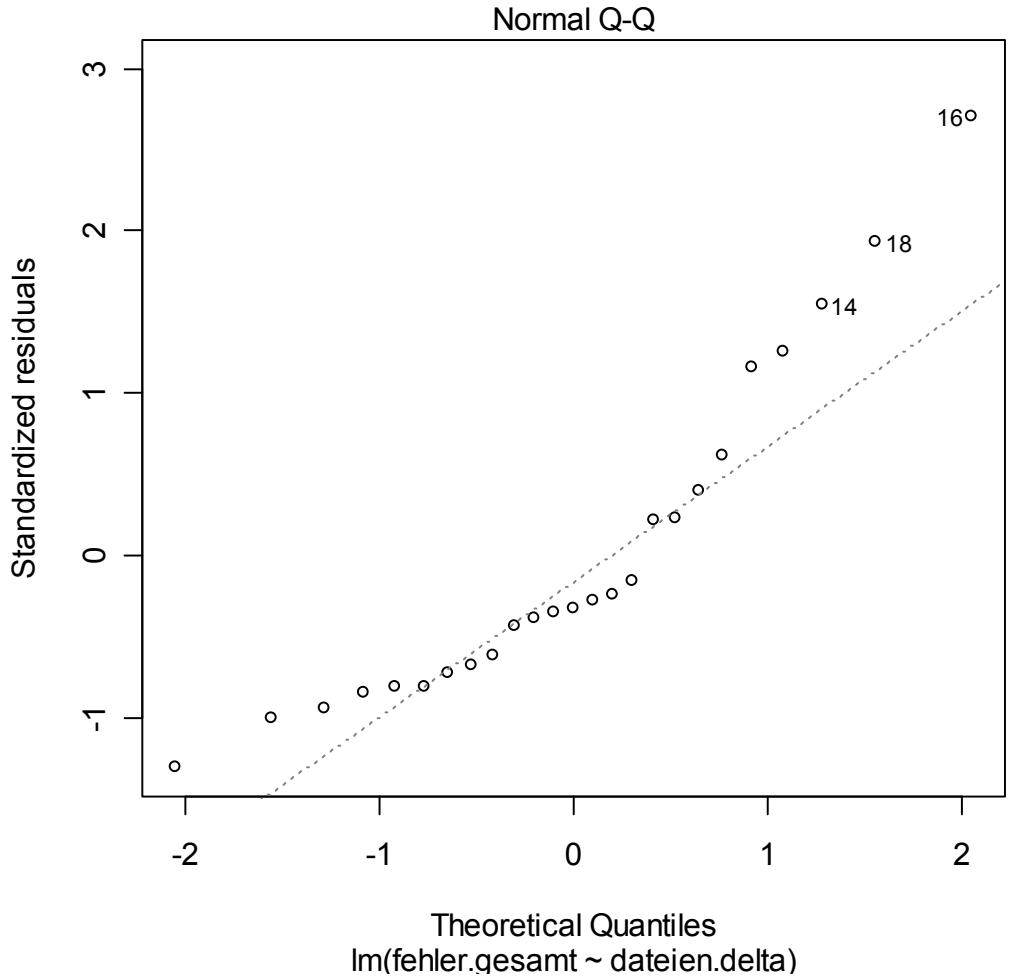

Abbildung 4-10: QQ-Plot Gesamtanzahl aller Feldfehler mit Δ-Anzahl-Dateien

Der QQ-Plot in Abbildung 4-10 mit Δ-Anzahl-Dateien als erklärender Variable ist nahezu ident mit dem QQ-Plot der linearen Regression mit Δ-Anzahl-Dateien und LOC als erklärender Variable (Abbildung 4-8). Auch in diesem Fall ist die Normalverteilungsannahme der Residuen annähernd erfüllt.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.gesamt
              Df Sum Sq Mean Sq F value    Pr(>F)
dateien.delta  1  55937   55937  36.783 3.473e-06 ***
Residuals     23  34976    1521
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA bestätigt die hohe Signifikanz von Δ-Anzahl-Dateien zur Erklärung der Gesamtfehleranzahl.

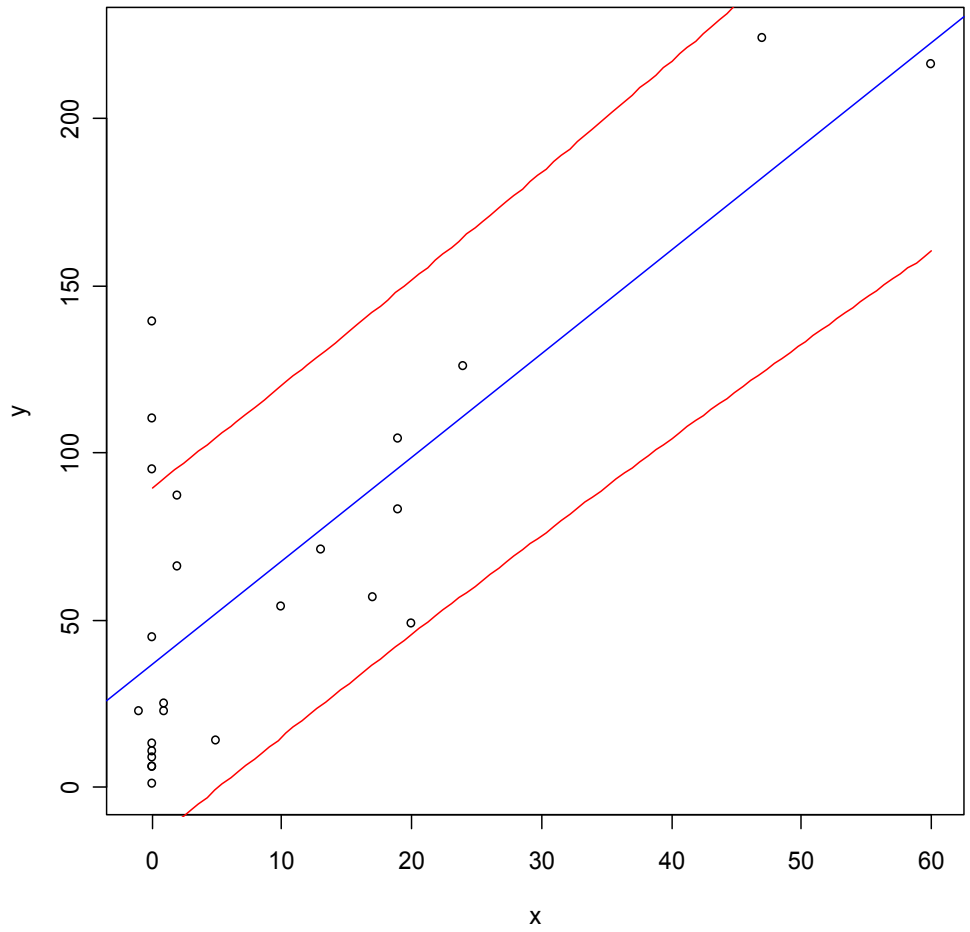

Abbildung 4-11: Prognoseintervall Gesamtanzahl aller Feldfehler mit Δ-Anzahl-Dateien

Das Prognoseintervall zum Niveau 80% in Abbildung 4-11 entspricht allerdings einer Schwankungsbreite von etwa 100 Gesamtfehlern. Wie aus dem Diagramm ersicht-lich, fallen drei Gesamtfehleranzahlen aus dem Konfidenzintervall heraus.

Eigentlich müsste analog zu Δ-Anzahl-Dateien auch Δ-LOC als quantitatives Änderungsmaß funktionieren, obwohl die lineare Regression mit allen Prediktoren als erklärende Variable keine relevante Signifikanz angezeigt hat. Zum Vergleich werden wir im Folgenden also auch noch Δ-LOC als erklärende Variable betrachten.

Lineare Regression mit Δ-LOC:

```
lm(formula = fehler.gesamt ~ loc.delta)

Residuals:
   Min     1Q Median     3Q    Max
-56.77 -31.65 -14.49  21.01  98.33

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.612e+01  1.131e+01   3.192 0.004053 **
loc.delta   4.359e-03  9.629e-04   4.527 0.000151 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 45.72 on 23 degrees of freedom
Multiple R-Squared: 0.4711,     Adjusted R-squared: 0.4482
F-statistic: 20.49 on 1 and 23 DF,  p-value: 0.0001515
```

Erstaunlicherweise fällt die Güte des Modells auf Basis von Δ-LOC im Vergleich zu Δ-Anzahl-Dateien stark ab. Ein Bestimmtheitsmaß von 0,4711 ist eigentlich nicht mehr akzeptabel.

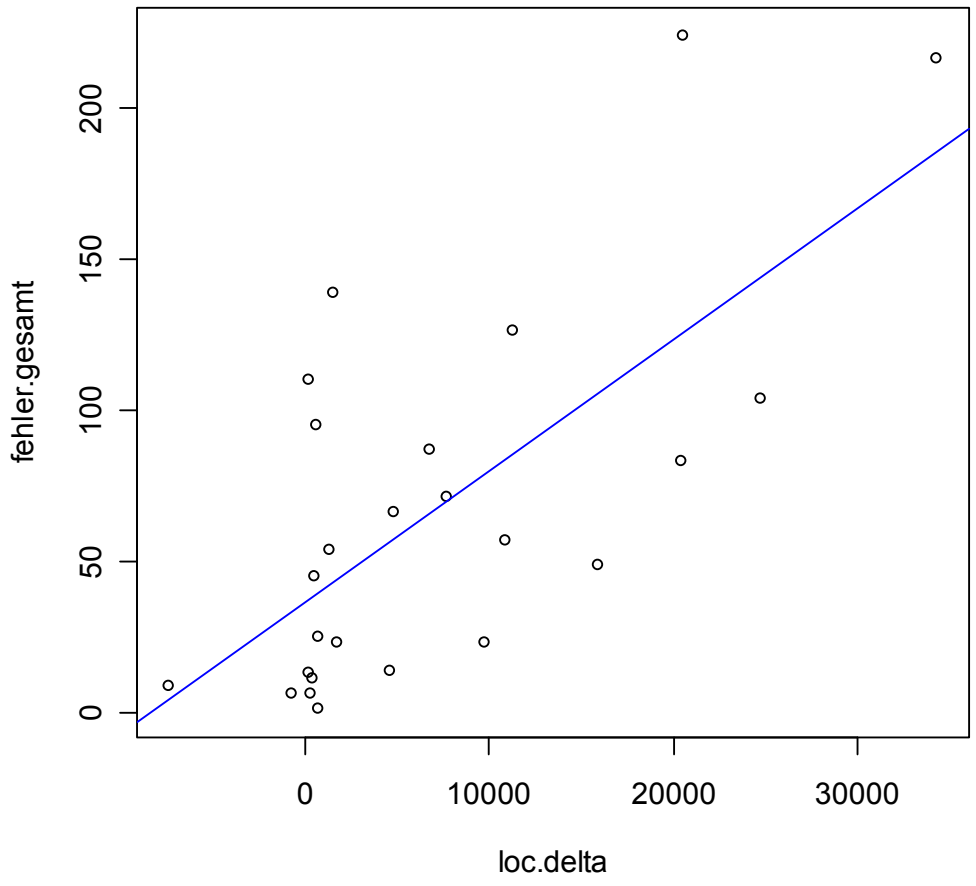

Abbildung 4-12: DOT-Plot Gesamtanzahl aller Feldfehler mit Δ-LOC

Im DOT-Plot der Abbildung 4-12 liegt der Achsenabschnitt der Regressionsgeraden bei 36,118 und damit sehr nahe dem Achsenabschnitt der linearen Regression mit Δ-Anzahl-Dateien. Der Achsenabschnitt ergibt sich auch hier, wie auch aus dem Diagramm ersichtlich, aus einer relativ breiten Streuung der Gesamtfehleranzahl bei Δ-LOC = 0.

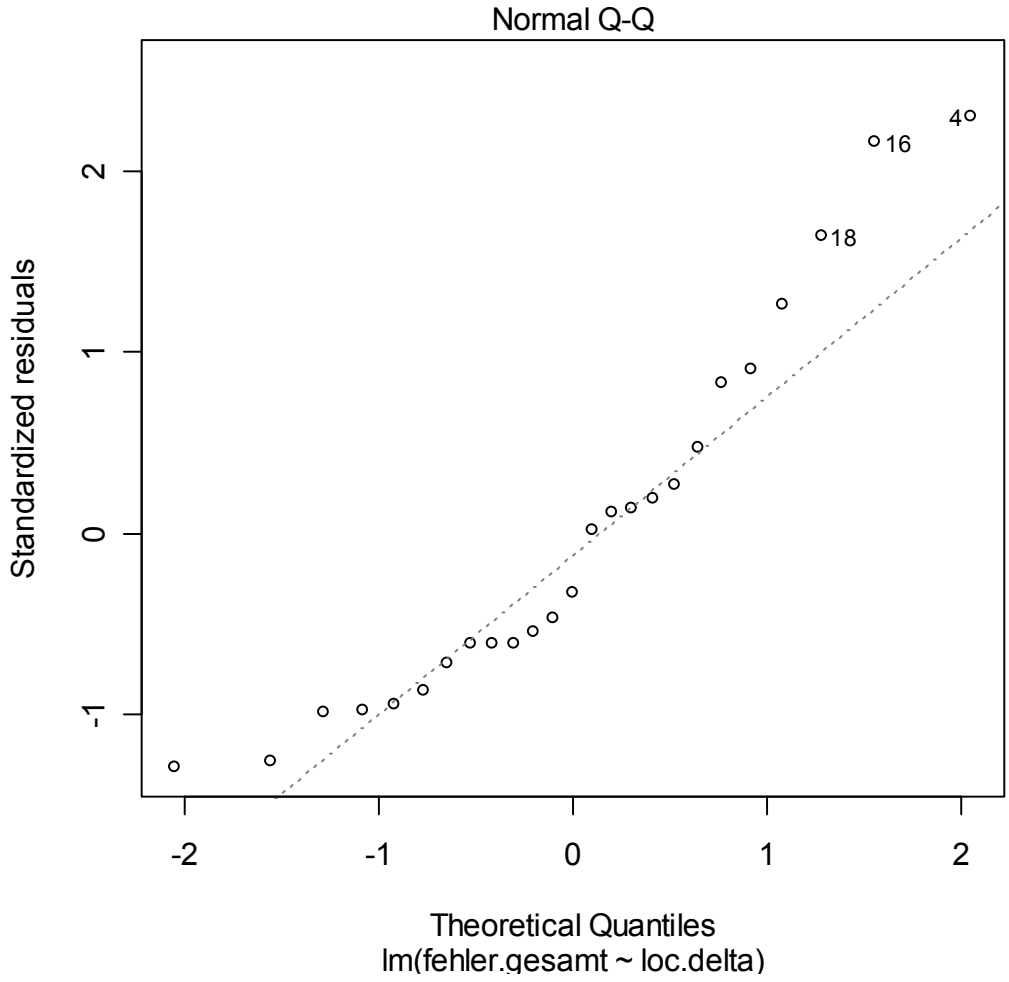

Abbildung 4-13: QQ-Plot Gesamtanzahl aller Feldfehler mit Δ-LOC

Der QQ-Plot in Abbildung 4-13 mit Δ-LOC als erklärende Variable ist ähnlich zu dem QQ-Plot der linearen Regression mit Δ-Anzahl-Dateien (Abbildung 4-10) bzw. Δ-Anzahl-Dateien und LOC (Abbildung 4-8) als erklärende Variable. Auch in diesem Fall ist die Normalverteilungsannahme der Residuen annähernd erfüllt.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.gesamt
           Df Sum Sq Mean Sq F value    Pr(>F)
loc.delta   1  42833   42833   20.49 0.0001515 ***
Residuals  23  48080    2090
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt jetzt auch eine hohe Signifikanz von Δ-LOC zur Erklärung der Gesamtfehleranzahl, allerdings ist diese nicht so hoch wie die Signifikanz von Δ-Anzahl-Dateien. Die p-Wert-Differenz zugunsten von Δ-Anzahl-Dateien beträgt 0,000148027. Offenbar liegt darin auch die Erklärung für die schlechtere Güte dieses Modells.

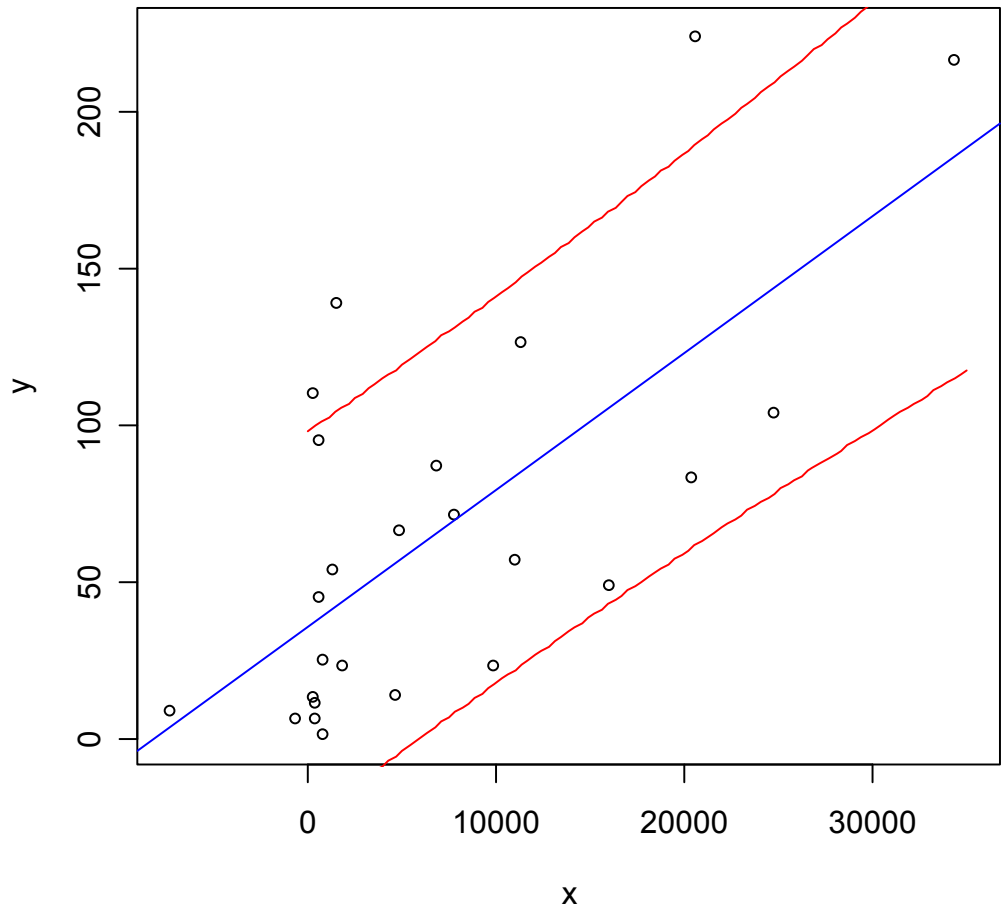

Abbildung 4-14: Prognoseintervall Gesamtanzahl aller Feldfehler mit Δ-LOC

Das Prognoseintervall zum Niveau 80% in Abbildung 4-14 entspricht auch hier noch einer Schwankungsbreite von etwa 100 Gesamtfehlern und unterscheidet sich kaum

vom Prognoseintervall mit Δ-Anzahl-Dateien als erklärende Variable (Abbildung 4-11). Die geringere Prediktoren-Signifikanz sowie die schlechtere Güte wirken sich am Prognoseintervall nicht merklich aus.

4.6.2 Detailanalyse der Anzahl Feldfehler mit Priorität 4

Die Modellparameter ergeben sich zu:

```
lm(formula = fehler.prio4 ~ loc + dateien.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-14.905  -6.361  -2.986   2.294  24.719

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    6.783e+00  1.069e+01   0.634    0.532
loc           -2.330e-06  4.918e-05  -0.047    0.963
dateien.delta  7.830e-01  1.464e-01   5.350 2.27e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.42 on 22 degrees of freedom
Multiple R-Squared: 0.6004,     Adjusted R-squared: 0.5641
F-statistic: 16.53 on 2 and 22 DF,  p-value: 4.148e-05
```

Auch die Güte dieses Modells ist verglichen mit dem Modell auf Basis aller Prediktoren ein wenig schlechter, aber noch im oberen Bereich der bisher erzielten Ergebnisse.

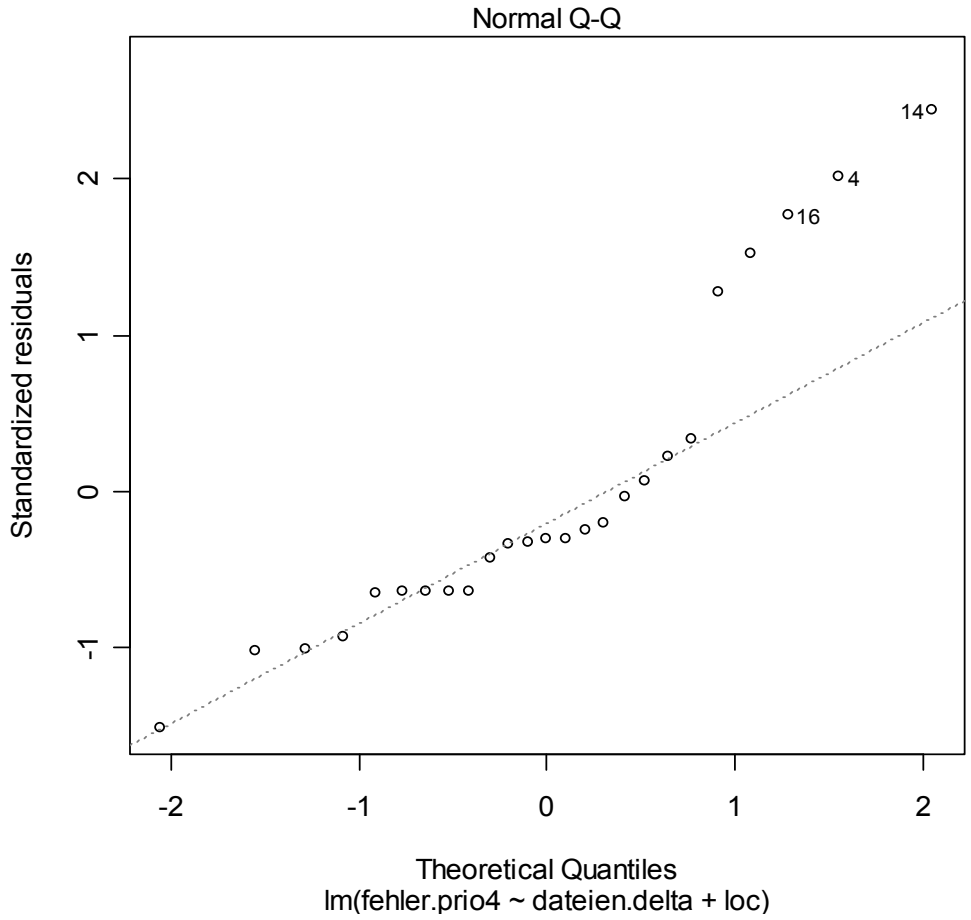

Abbildung 4-15: QQ-Plot Anzahl Feldfehler mit Priorität 4 mit 2 Prediktoren

Der QQ-Plot in Abbildung 4-15 zeigt, dass von den 25 theoretischen Residuen 5 nicht in der Nähe der Standardverteilungsgeraden liegen. Die restlichen theoretischen Residuen liegen annähernd auf der Geraden, d.h. für einen Großteil der theoretischen Residuen ist die für die lineare Regression notwendige Normalverteilungsannahme der Residuen erfüllt. Der QQ-Plot ist auch vergleichbar mit dem QQ-Plot des Modells Anzahl Feldfehler mit Priorität 4 mit allen Prediktoren (Abbildung 4-7). Die Ergebnisse der Modellberechnungen sind daher gerade im wichtigen Bereich der Feldfehler mit Priorität 4 mit Vorsicht zu interpretieren.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio4
              Df Sum Sq Mean Sq F value    Pr(>F)
dateien.delta  1 3589.0  3589.0 33.0530 8.776e-06 ***
loc            1    0.2     0.2  0.0022    0.9626
Residuals     22 2388.8   108.6
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die Signifikanz von Δ-Anzahl-Dateien ist hier noch immer sehr hoch, während die Signifikanz von LOC deutlich abgenommen hat. Zur einfacheren graphischen Darstellung bzw. Analyse betrachten wir im Folgenden daher die lineare Regression wieder nur mehr mit der erklärenden Variablen Δ-Anzahl-Dateien.

Lineare Regression mit Δ-Anzahl-Dateien:

```
lm(formula = fehler.prio4 ~ dateien.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-15.001  -6.290  -3.076   2.356  24.710

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     6.2903     2.4056   2.615   0.0155 *
dateien.delta   0.7855     0.1336   5.878 5.44e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.19 on 23 degrees of freedom
Multiple R-Squared: 0.6004,     Adjusted R-squared: 0.583
F-statistic: 34.55 on 1 and 23 DF,  p-value: 5.438e-06
```

Die Güte des Modells auf Basis Δ-Anzahl-Dateien hat exakt den gleichen Wert wie das Modell auf Basis zweier Prediktoren.

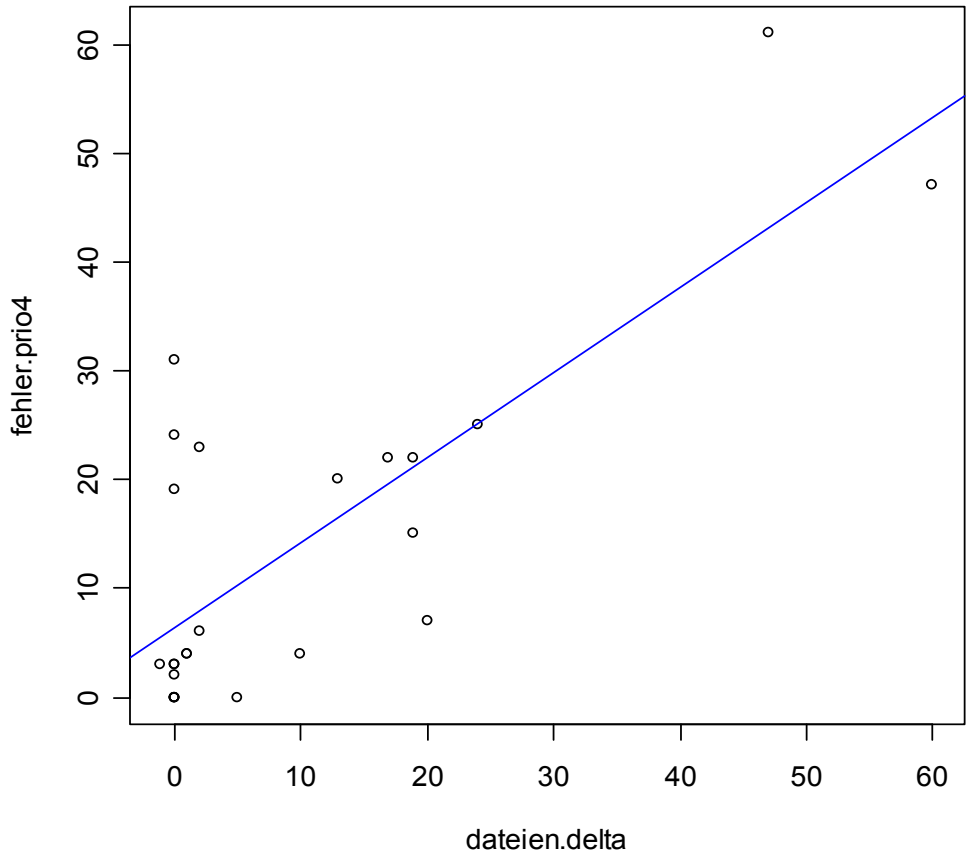

Abbildung 4-16: DOT-Plot Anzahl Feldfehler mit Priorität 4 mit Δ-Anzahl-Dateien

Der Achsenabschnitt der Regressionsgeraden im DOT-Plot der Abbildung 4-16 liegt – trotz einer relativ breiten Streuung der Anzahl Feldfehler mit Priorität 4 bei Δ-Anzahl-Dateien = 0 – bei 6,29 und damit schon sehr nahe beim Nullpunkt.

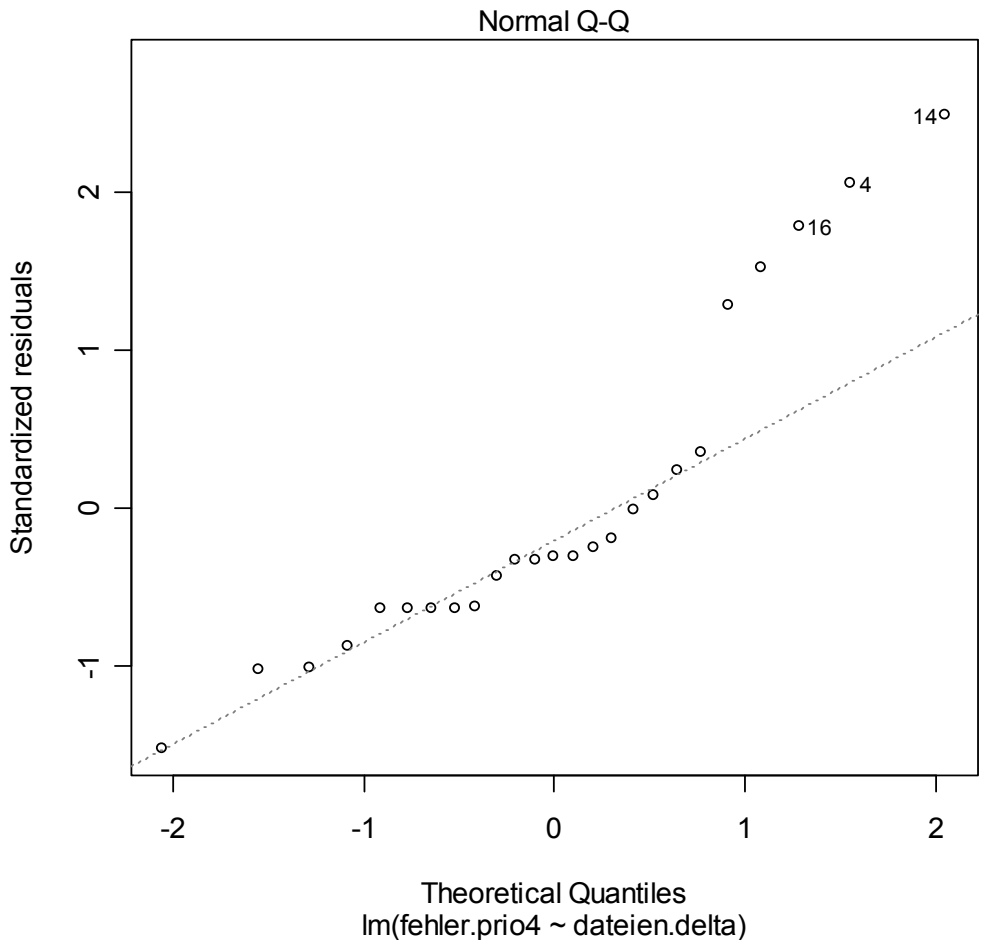

Abbildung 4-17: QQ-Plot Anzahl Feldfehler mit Priorität 4 mit Δ-Anzahl-Dateien

Der QQ-Plot in Abbildung 4-17 mit Δ-Anzahl-Dateien als erklärende Variable ist wieder nahezu ident mit dem QQ-Plot der linearen Regression mit Δ-Anzahl-Dateien und LOC als erklärende Variable (Abbildung 4-15). Auch in diesem Fall ist die Normalverteilungsannahme der Residuen nur bis auf 5 Ausreißer annähernd erfüllt und damit das Ergebnis aus dieser Warte mit Vorsicht zu interpretieren.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio4
              Df Sum Sq Mean Sq F value    Pr(>F)
dateien.delta  1 3589.0  3589.0  34.552 5.438e-06 ***
Residuals     23 2389.0   103.9
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt die hohe Signifikanz von Δ-Anzahl-Dateien zur Erklärung der Anzahl Feldfehler mit Priorität 4.

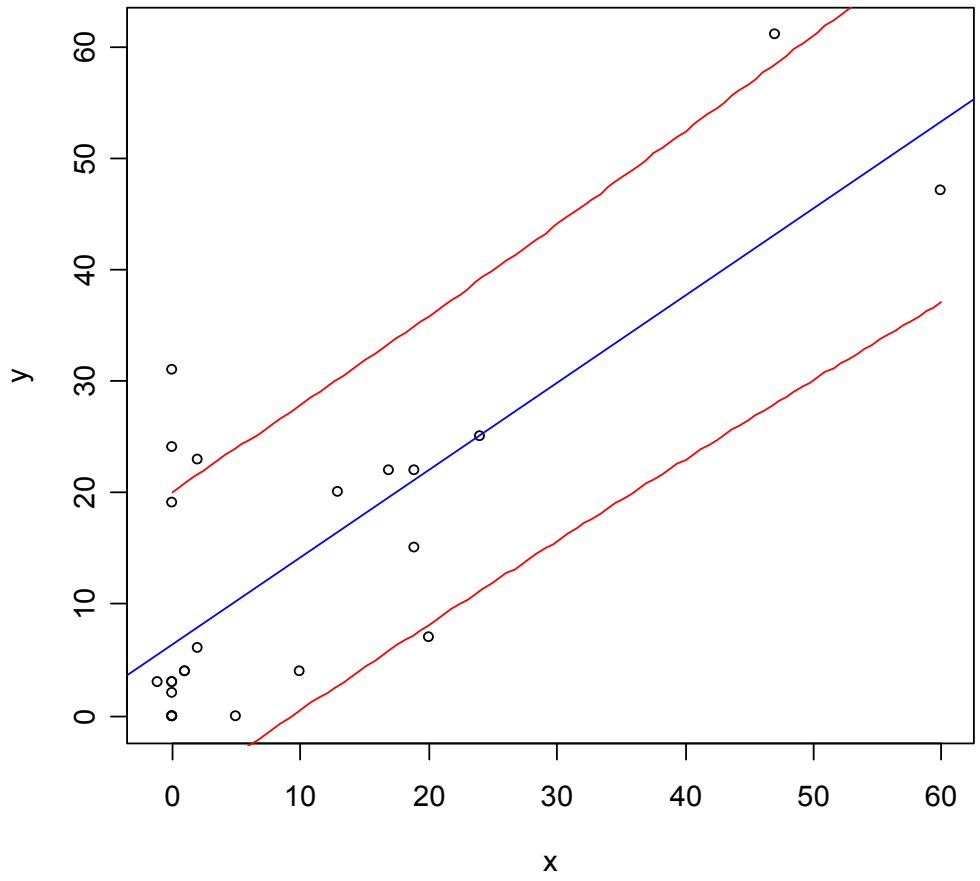

Abbildung 4-18: Prognoseintervall Anzahl Feldfehler mit Priorität 4 mit Δ-Anzahl-Dateien

Das Prognoseintervall zum Niveau 80% in Abbildung 4-18 entspricht einer Schwankungsbreite von etwa ± 10 Fehlern der Priorität 4. Wie aus dem Diagramm ersicht-lich, fallen 5 gemessene Fehleranzahlen der Priorität 4 aus dem Prognoseintervall heraus.

Wie auch bei der Gesamtfehleranzahl müsste analog zu Δ-Anzahl-Dateien auch Δ-LOC als quantitatives Änderungsmaß funktionieren, obwohl die lineare Regression mit allen Prediktoren als erklärende Variable keine relevante Signifikanz angezeigt hat. Zum Vergleich erfolgt also auch hier noch eine lineare Regression mit Δ-LOC als erklärende Variable.

Lineare Regression mit Δ-LOC:

```
lm(formula = fehler.prio4 ~ loc.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-17.622  -6.767  -3.950   5.302  32.851

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 6.512048   3.042733   2.140 0.043172 *
loc.delta   0.001053   0.000259   4.067 0.000476 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.3 on 23 degrees of freedom
Multiple R-Squared: 0.4183,     Adjusted R-squared: 0.393
F-statistic: 16.54 on 1 and 23 DF,  p-value: 0.0004761
```

Wie auch schon beim Modell der Gesamtzahl aller Feldfehler auf Basis ausgesuchter Prediktoren fällt die Güte bei Verwendung von Δ-LOC als Prediktor stark ab.

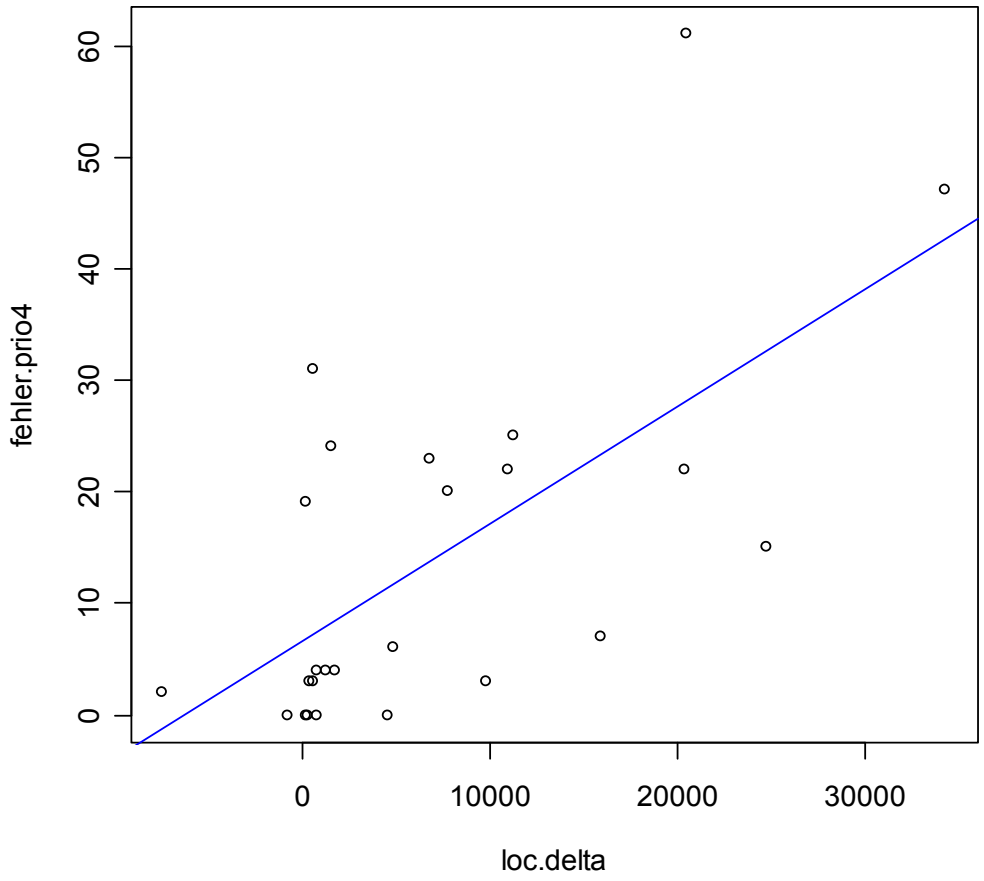

Abbildung 4-19: DOT-Plot Anzahl Feldfehler mit Priorität 4 mit Δ-LOC

Der Achsenabschnitt der Regressionsgeraden im DOT-Plot der Abbildung 4-19 liegt bei 6,512 und damit sehr nahe dem Achsenabschnitt der linearen Regression mit Δ-Anzahl-Dateien.

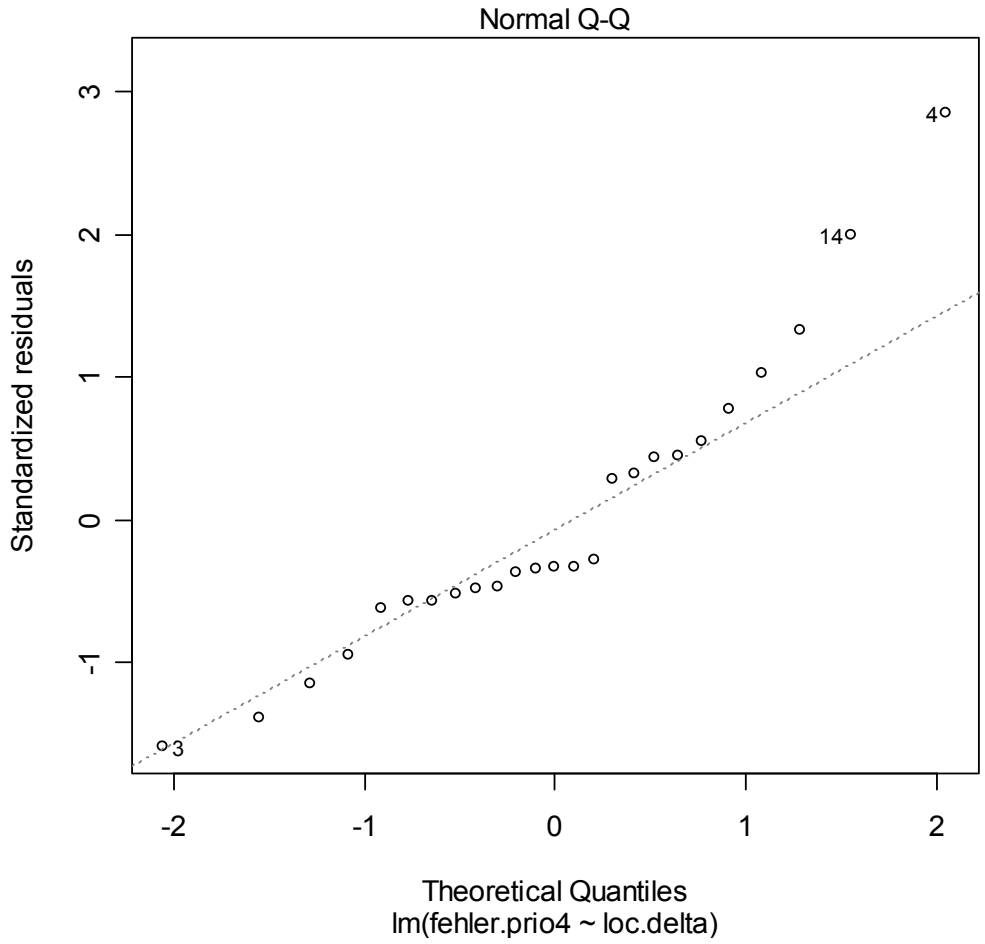

Abbildung 4-20: QQ-Plot Anzahl Feldfehler mit Priorität 4 mit Δ-LOC

Der QQ-Plot in Abbildung 4-20 mit Δ-LOC als erklärende Variable erfüllt die Normalverteilungsannahme der Residuen allerdings ein wenig besser als mit der erklärenden Variablen Δ-Anzahl-Dateien.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio4
           Df Sum Sq Mean Sq F value    Pr(>F)
loc.delta   1 2500.8  2500.8  16.542 0.0004761 ***
Residuals  23 3477.2   151.2
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt jetzt auch eine hohe Signifikanz von Δ-LOC zur Erklärung der Gesamtfehleranzahl, allerdings ist diese nicht so hoch wie die Signifikanz von Δ-Anzahl-Dateien. Die p-Wert-Differenz zugunsten von Δ-Anzahl-Dateien beträgt 0,000470662. Auch hier dürfte darin die Erklärung für die schlechtere Güte des Modells auf Basis Δ-LOC liegen.

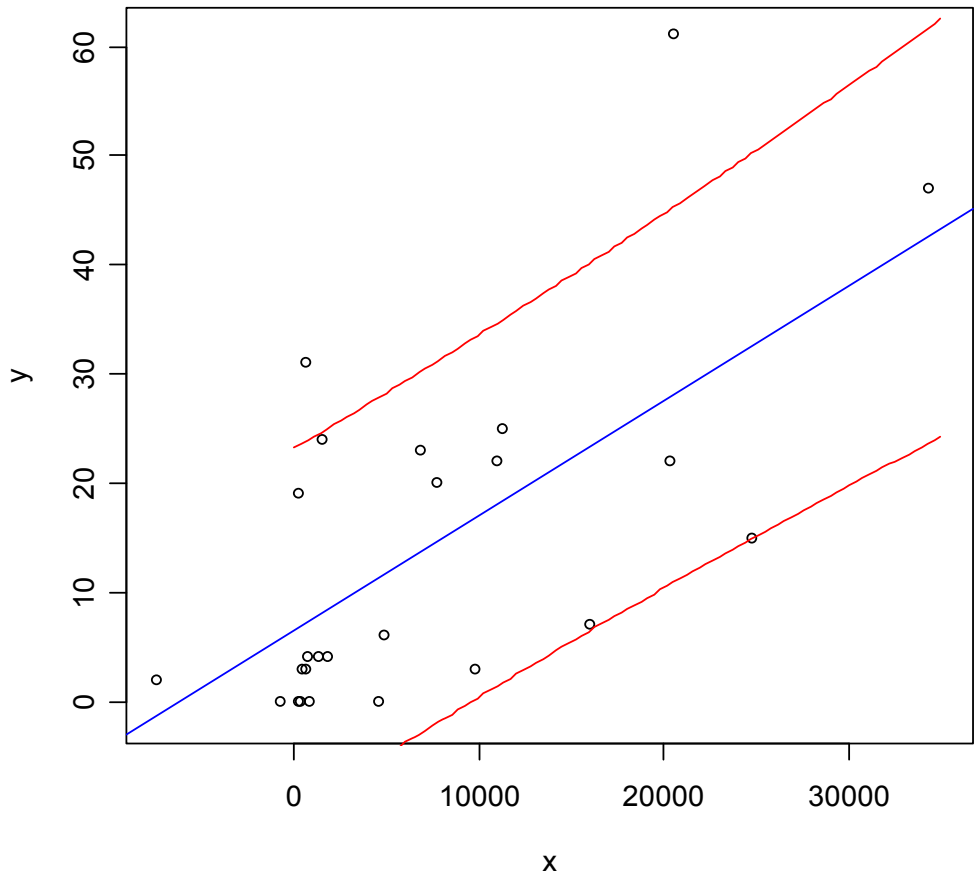

Abbildung 4-21: Prognoseintervall Anzahl Feldfehler mit Priorität 4 mit Δ-LOC

Das Prognoseintervall zum Niveau 80% in Abbildung 4-21 entspricht auch hier einer Schwankungsbreite von etwa ± 10 Gesamtfehlern und unterscheidet sich trotz schlechterem Bestimmtheitsmaß kaum vom Prognoseintervall mit Δ-Anzahl-Dateien als erklärender Variablen (Abbildung 4-18).

4.6.3 Detailanalyse der Anzahl Feldfehler mit Priorität 3

Die Modellparameter ergeben sich zu:

```
lm(formula = fehler.prio3 ~ loc + dateien.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-21.161 -12.444  -2.539   6.051  38.094

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -8.816e+00  1.771e+01  -0.498    0.624
loc            1.232e-04  8.144e-05   1.513    0.145
dateien.delta  1.322e+00  2.424e-01   5.456 1.76e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.26 on 22 degrees of freedom
Multiple R-Squared: 0.5768,     Adjusted R-squared: 0.5384
F-statistic: 14.99 on 2 and 22 DF,  p-value: 7.792e-05
```

Auch dieses Modell hat eine schlechtere Güte als das vergleichbare Modell auf Basis aller Prediktoren. Die Differenz der Bestimmheitsmaße liegt dabei im Bereich der bisher festgestellten Unterschiede.

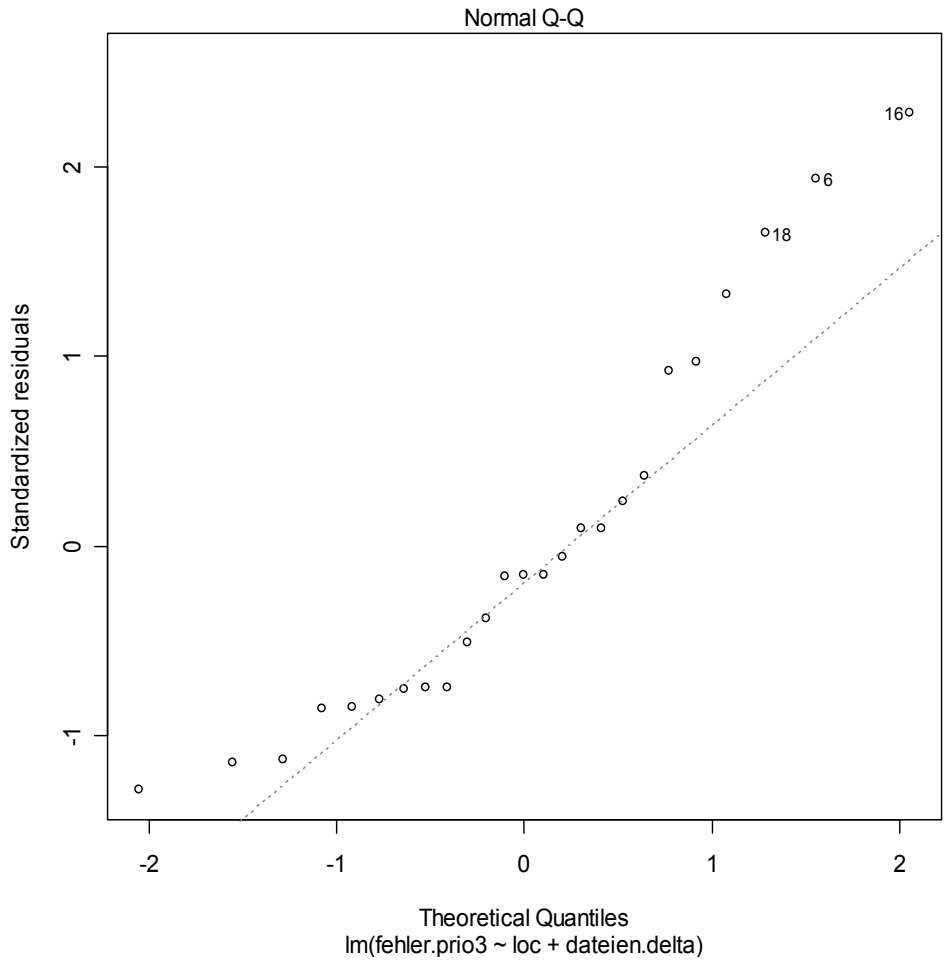

Abbildung 4-22: QQ-Plot Anzahl Feldfehler mit Priorität 3 mit 2 Prediktoren

Im QQ-Plot der Abbildung 4-22 ist die für die lineare Regression notwendige Normalverteilungsannahme der Residuen annähernd erfüllt. Vergleicht man mit dem QQ-Plot des Modells Anzahl Feldfehler mit Priorität 3 mit allen Prediktoren (Abbildung 4-6), kann man erkennen, dass das Modell mit allen Prediktoren eine noch bessere Übereinstimmung mit der Normalverteilungsannahme bietet.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio3
              Df Sum Sq Mean Sq F value    Pr(>F)
loc            1   66.9    66.9  0.2247    0.6402
dateien.delta  1 8862.3  8862.3 29.7641 1.761e-05 ***
Residuals     22 6550.5   297.8
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die Signifikanz von Δ-Anzahl-Dateien ist hier noch immer sehr hoch, während die Signifikanz von LOC eigentlich nicht mehr vorhanden ist. Zur einfacheren

graphischen Darstellung bzw. Analyse betrachten wir im Folgenden daher die lineare Regression nur mehr mit der erklärenden Variablen Δ-Anzahl-Dateien.

Lineare Regression mit Δ-Anzahl-Dateien:

```
lm(formula = fehler.prio3 ~ dateien.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-17.500 -14.256  -7.065  10.744  40.744

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    17.2557     4.1855   4.123 0.000415 ***
dateien.delta   1.1908     0.2325   5.122 3.45e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.73 on 23 degrees of freedom
Multiple R-Squared: 0.5328,     Adjusted R-squared: 0.5125
F-statistic: 26.23 on 1 and 23 DF,  p-value: 3.452e-05
```

Das Bestimmtheitsmaß dieses Modells entspricht mit 0,5328 im Wesentlichen dem Bestimmtheitsmaß des Modells auf Basis zweier Prediktoren.

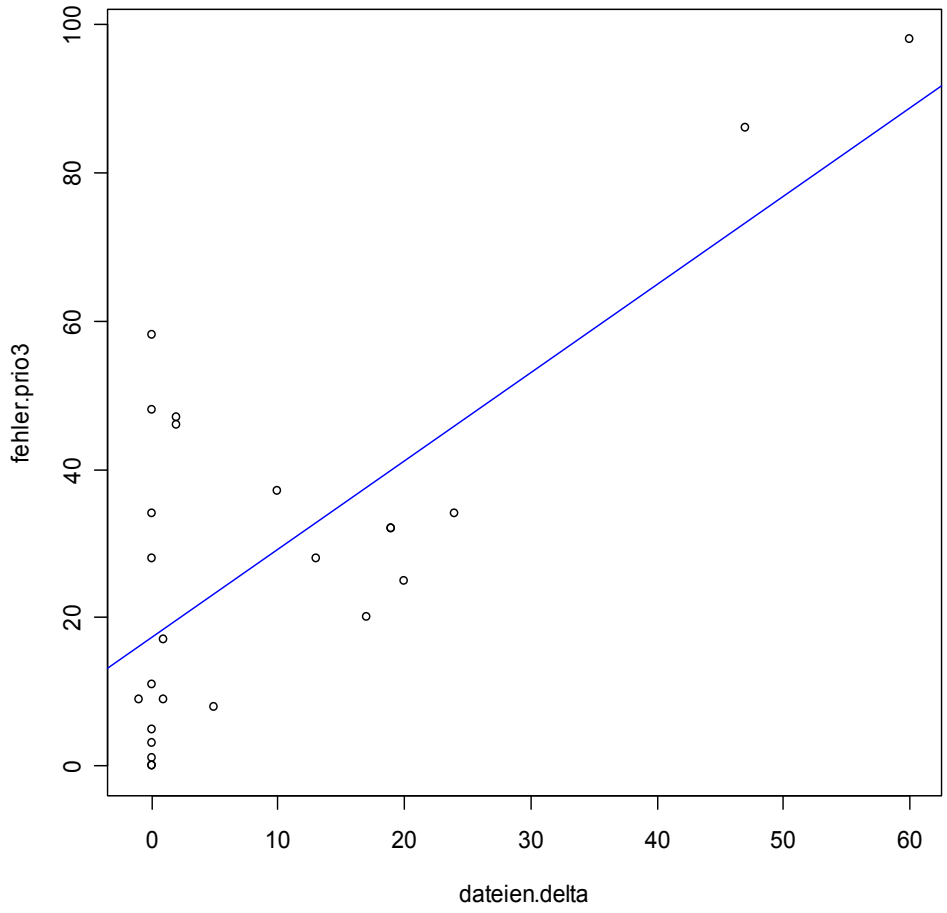

Abbildung 4-23: DOT-Plot Anzahl Feldfehler mit Priorität 3 mit Δ-Anzahl-Dateien

Im DOT-Plot der Abbildung 4-23 liegt der Achsenabschnitt der Regressionsgeraden bei 17,256 und ergibt sich, wie auch aus dem Diagramm ersichtlich, aus einer relativ breiten Streuung der Anzahl Feldfehler mit Priorität 3 bei Δ-Anzahl-Dateien = 0.

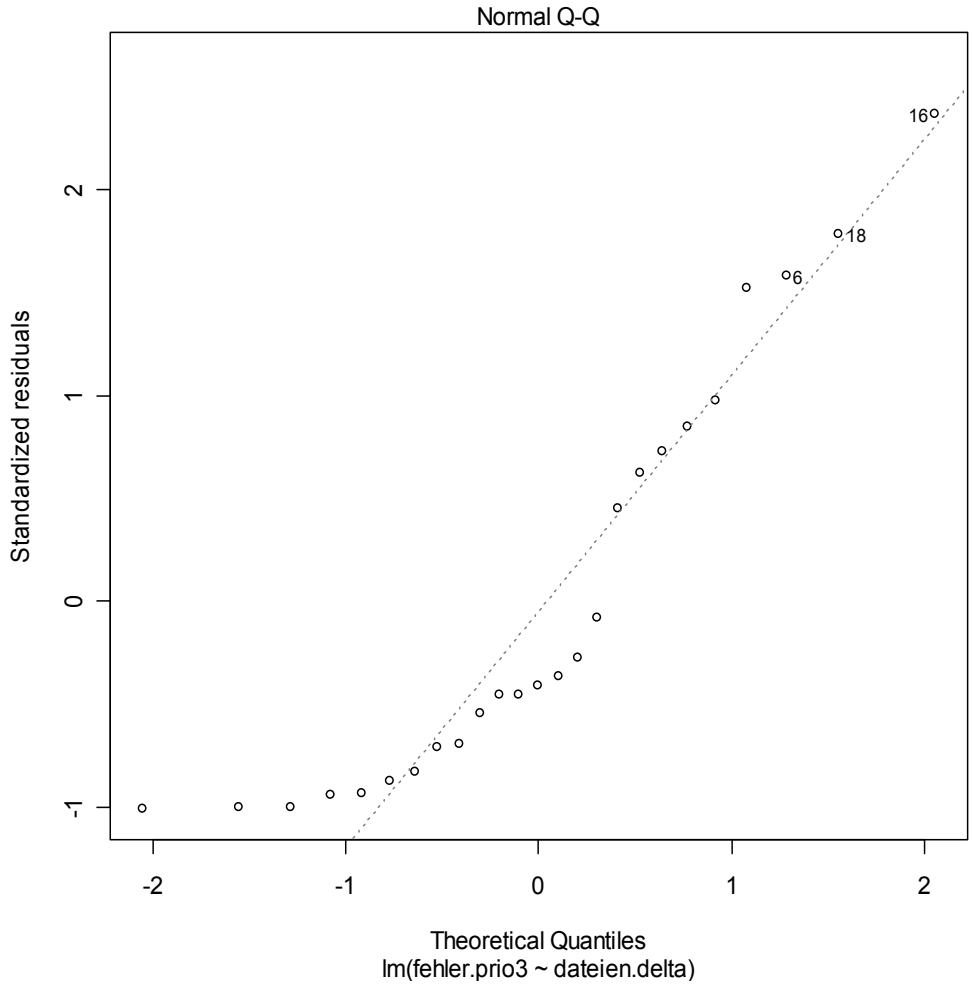

Abbildung 4-24: QQ-Plot Anzahl Feldfehler mit Priorität 3 mit Δ-Anzahl-Dateien

Der QQ-Plot in Abbildung 4-24 mit Δ-Anzahl-Dateien als erklärende Variable zeigt mit Ausnahme von 3 Ausreißern eine gute Übereinstimmung mit der Normalverteilungsannahme der Residuen. Verglichen mit dem QQ-Plot mit Δ-Anzahl-Dateien und LOC als Prediktoren (Abbildung 4-22) zeigt sich zwar eine andere Verteilung der Residuen, im Durchschnitt ist die Abweichung aber als ähnlich zu betrachten.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio3
              Df Sum Sq Mean Sq F value    Pr(>F)
dateien.delta  1 8247.7  8247.7   26.23 3.452e-05 ***
Residuals     23 7232.0   314.4
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt die hohe Signifikanz von Δ-Anzahl-Dateien zur Erklärung der Anzahl Feldfehler mit Priorität 3.

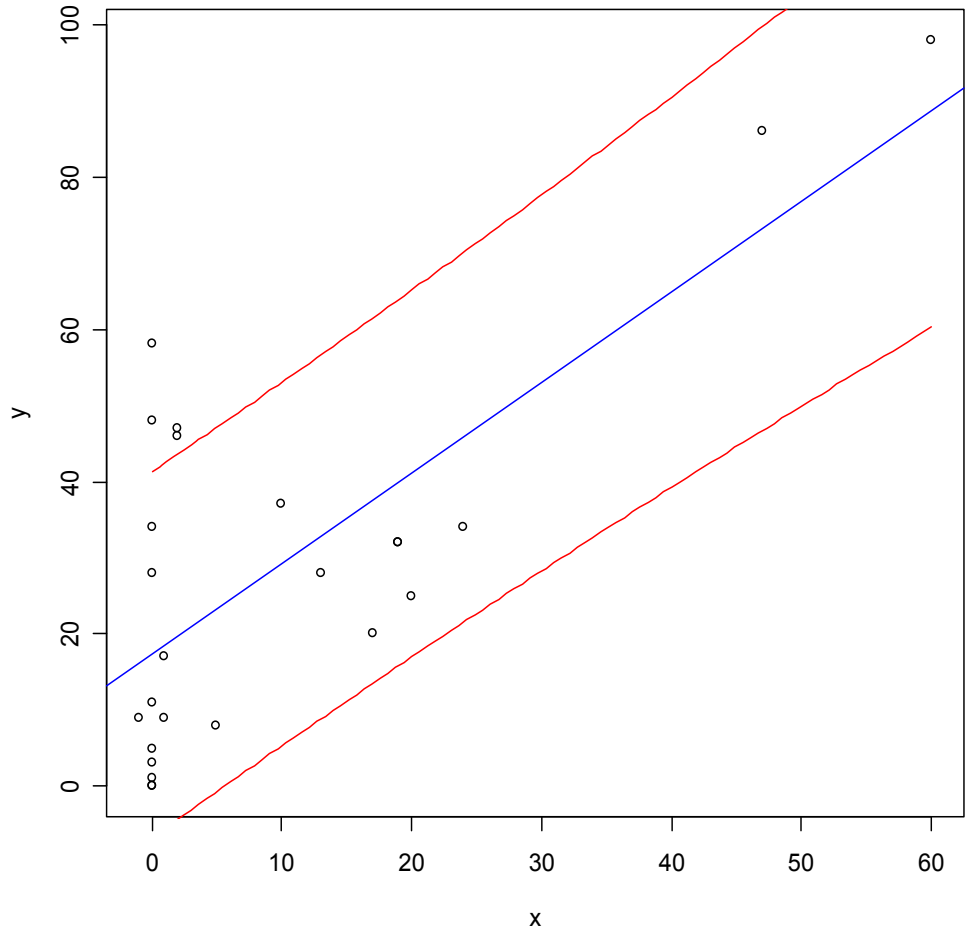

Abbildung 4-25: Prognoseintervall Anzahl Feldfehler mit Priorität 3 mit Δ-Anzahl-Dateien

Das Prognoseintervall zum Niveau 80% in Abbildung 4-25 entspricht einer Schwankungsbreite von etwa ± 20 Fehlern der Priorität 3. Wie aus dem Diagramm ersicht-lich fallen 4 gemessene Fehleranzahlen der Priorität 3 aus dem Prognoseintervall heraus.

Zum Vergleich auch hier wieder die lineare Regression mit Δ-LOC als erklärende Variable.

Lineare Regression mit Δ-LOC:

```
lm(formula = fehler.prio3 ~ loc.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-26.113 -16.879  -2.049  17.468  38.220

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.723e+01  4.989e+00   3.453  0.00216 **
loc.delta   1.649e-03  4.246e-04   3.884  0.00075 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.16 on 23 degrees of freedom
Multiple R-Squared: 0.3961,     Adjusted R-squared: 0.3698
F-statistic: 15.09 on 1 and 23 DF,  p-value: 0.00075
```

Obwohl sich Δ-LOC genauso wie Δ-Anzahl-Dateien als Prediktor eignen müsste, liegt die Güte dieses Modells unter 0,4 ist daher nicht mehr akzeptabel.

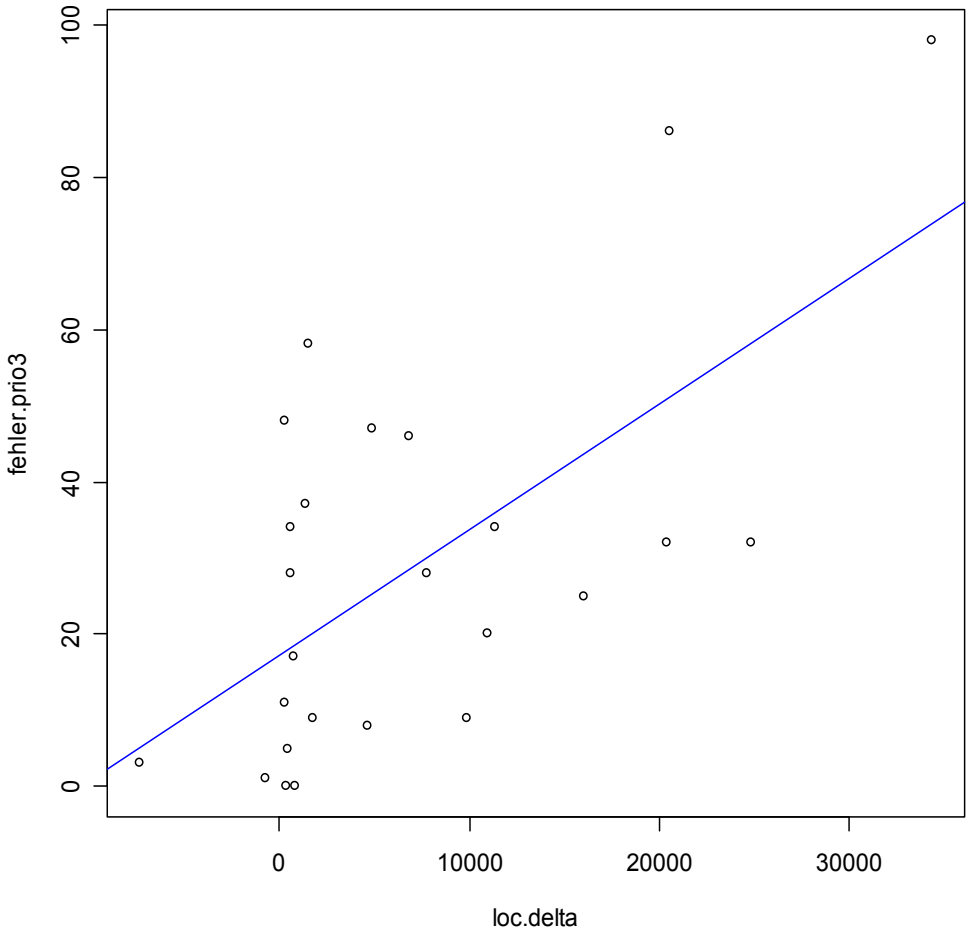

Abbildung 4-26: DOT-Plot Anzahl Feldfehler mit Priorität 3 mit Δ-LOC

Der Achsenabschnitt der Regressionsgeraden im DOT-Plot der Abbildung 4-26 liegt bei 17,228 und damit sehr nahe dem Achsenabschnitt der linearen Regression mit Δ-Anzahl-Dateien. Der Achsenabschnitt ergibt sich auch hier, wie aus dem Diagramm ersichtlich, aus einer relativ breiten Streuung der Anzahl Feldfehler mit Priorität 3 bei Δ-LOC = 0.

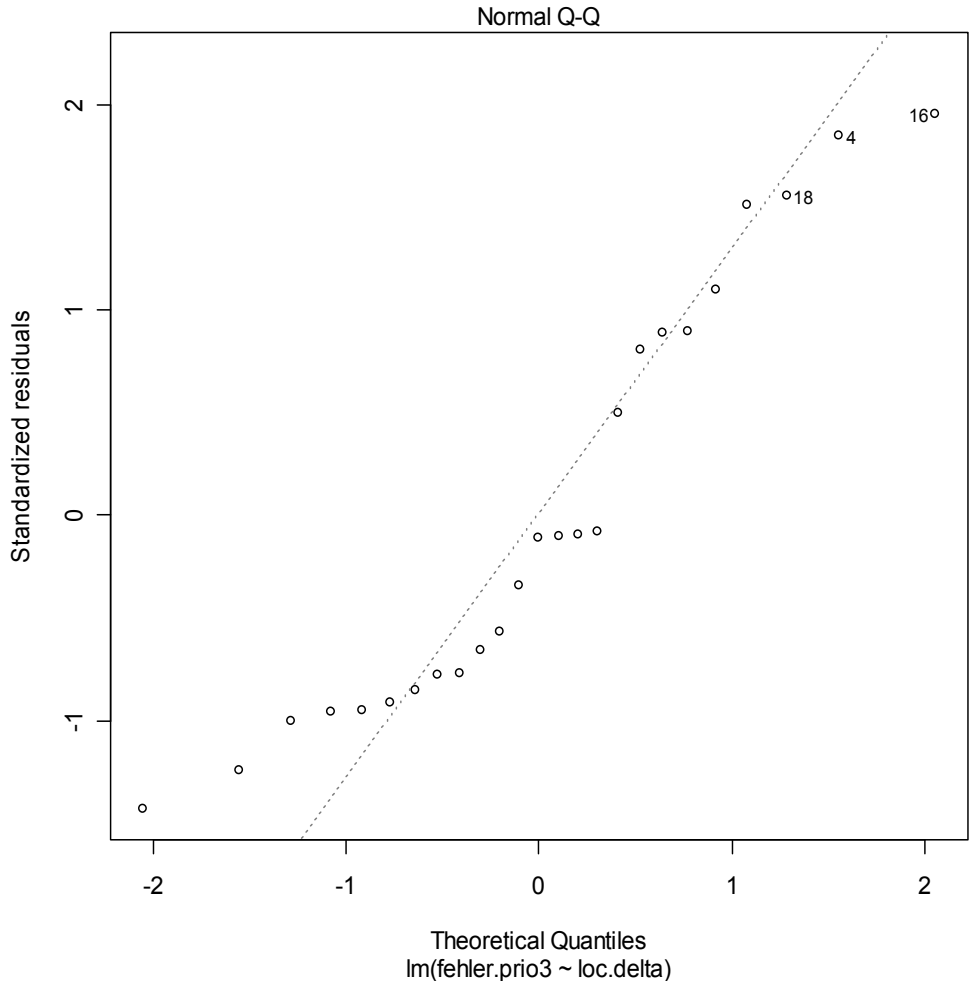

Abbildung 4-27: QQ-Plot Anzahl Feldfehler mit Priorität 3 mit Δ-LOC

Der QQ-Plot in Abbildung 4-27 mit Δ-LOC als erklärende Variable ist ähnlich dem QQ-Plot der linearen Regression mit Δ-Anzahl-Dateien (Abbildung 4-24) bzw. Δ-Anzahl-Dateien und LOC (Abbildung 4-22) als erklärende Variable. Auch in diesem Fall ist die Normalverteilungsannahme der Residuen annähernd erfüllt.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio3
           Df Sum Sq Mean Sq F value  Pr(>F)
loc.delta   1 6131.5  6131.5  15.086 0.00075 ***
Residuals 23 9348.2   406.4
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt jetzt auch eine hohe Signifikanz von Δ-LOC zur Erklärung der Gesamtfehleranzahl, allerdings ist diese nicht so hoch wie die Signifikanz von Δ-Anzahl-Dateien. Die p-Wert-Differenz zugunsten von Δ-Anzahl-Dateien beträgt 0,00071548.

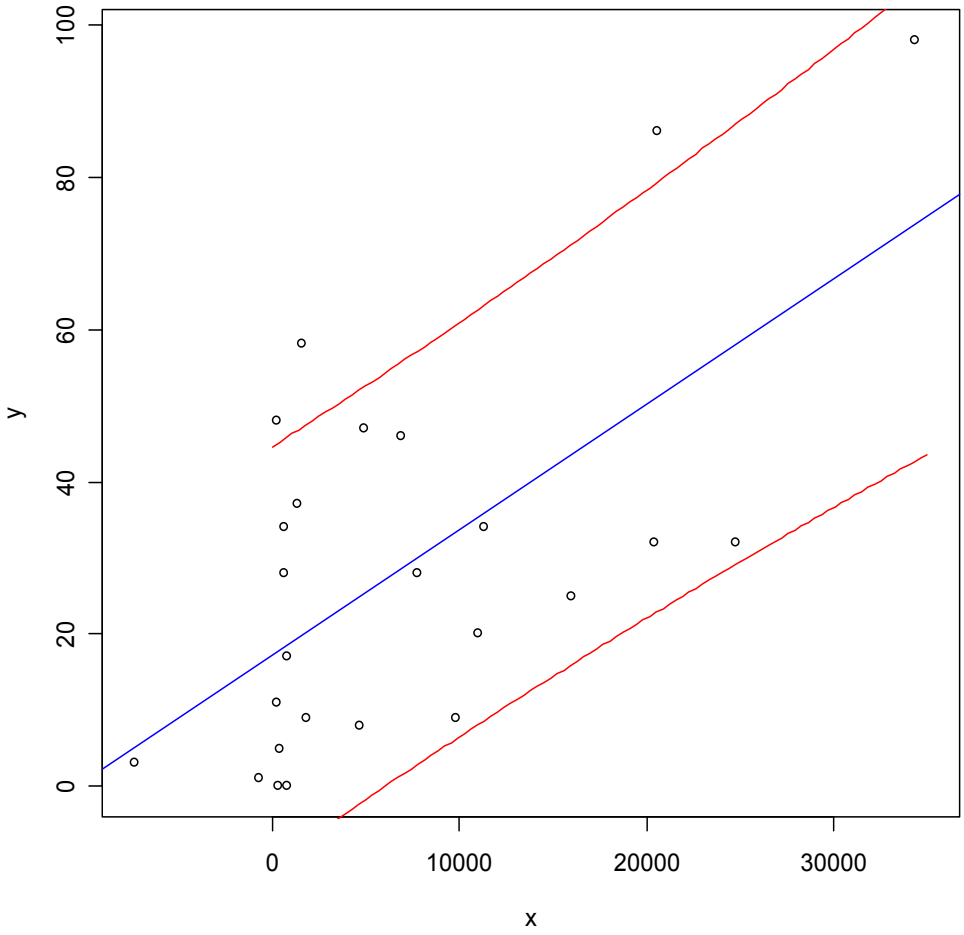

Abbildung 4-28: Konfidenzintervall Anzahl Feldfehler mit Priorität 3 mit Δ-LOC

Das Konfidenzintervall zum Niveau 80% in Abbildung 4-28 entspricht trotz des inakzeptablen Bestimmtheitsmaßes auch hier einer Schwankungsbreite von etwa ±

20 Gesamtfehlern und unterscheidet sich erstaunlicherweise kaum vom Prognoseintervall mit Δ-Anzahl-Dateien als erklärende Variable.

4.6.4 Detailanalyse der Anzahl Feldfehler mit Priorität 2

Die Modellparameter ergeben sich zu:

```
lm(formula = fehler.prio2 ~ loc + dateien.delta)

Residuals:
   Min     1Q Median     3Q    Max
-7.888 -4.788 -1.655  3.374 20.776

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.150e+00  7.249e+00  -0.159 0.875384
loc            4.021e-05  3.334e-05   1.206 0.240575
dateien.delta  4.363e-01  9.921e-02   4.398 0.000228 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.064 on 22 degrees of freedom
Multiple R-Squared: 0.4699,     Adjusted R-squared: 0.4217
F-statistic: 9.749 on 2 and 22 DF,  p-value: 0.0009296
```

Die Güte dieses Modells war schon auf Basis aller Prediktoren nicht sehr gut (0,5065) und liegt nun mit 0,4699 eigentlich schon im nicht akzeptablen Bereich.

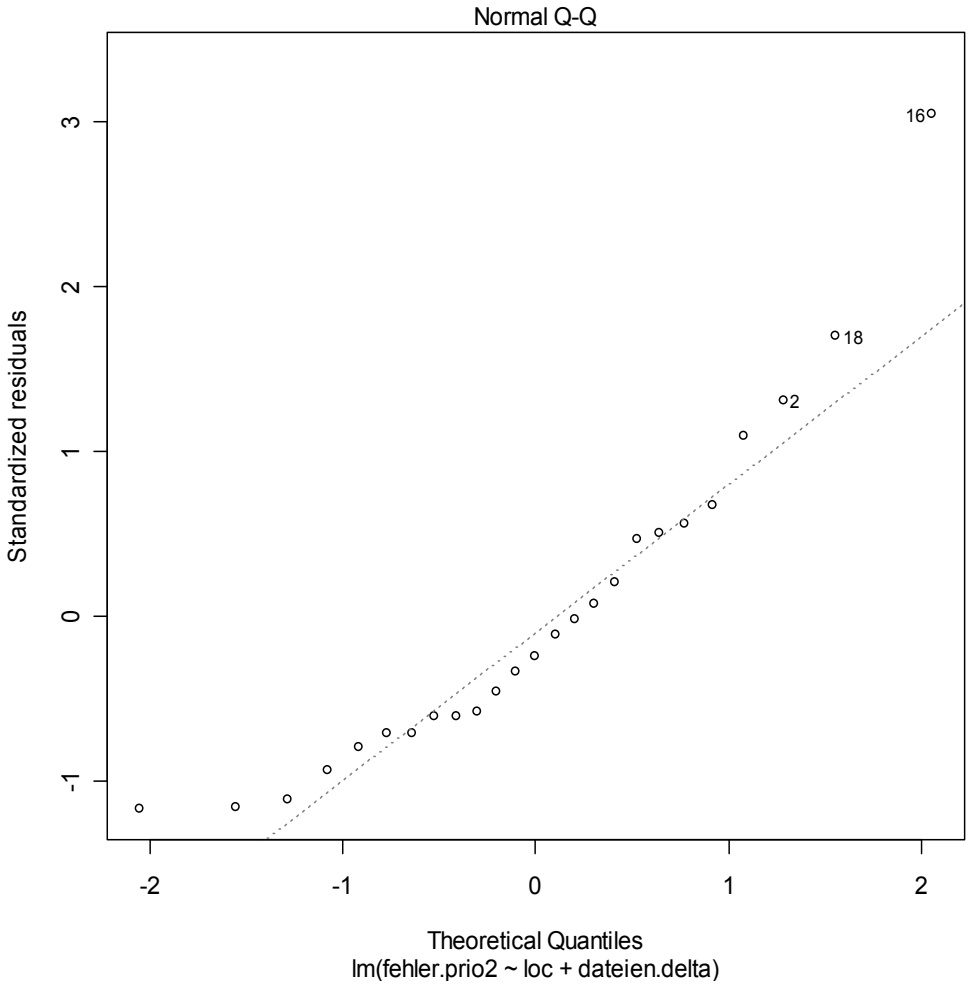

Abbildung 4-29: QQ-Plot Anzahl Feldfehler mit Priorität 2 mit 2 Prediktoren

Im QQ-Plot der Abbildung 4-29 ist die für die lineare Regression notwendige Normalverteilungsannahme der Residuen annähernd erfüllt. Das Modell Anzahl Feldfehler mit Priorität 2 mit allen Prediktoren (Abbildung 4-5) zeigt aber eine noch bessere Übereinstimmung der Residuen mit der Normalverteilungsannahme.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio2
              Df  Sum Sq Mean Sq F value    Pr(>F)
loc            1    7.84    7.84  0.1572 0.6955861
dateien.delta  1  965.07  965.07 19.3412 0.0002283 ***
Residuals     22 1097.73   49.90
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die Signifikanz von Δ-Anzahl-Dateien ist hier noch immer sehr hoch, während die Signifikanz von LOC eigentlich nicht mehr vorhanden ist. Zur einfacheren

graphischen Darstellung bzw. Analyse betrachten wir im Folgenden daher die lineare Regression nur mehr mit der erklärenden Variablen Δ-Anzahl-Dateien.

Lineare Regression mit Δ-Anzahl-Dateien:

```
lm(formula = fehler.prio2 ~ dateien.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-10.834  -5.047  -1.359   3.150  21.641

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   7.35870    1.68372   4.371 0.000224 ***
dateien.delta 0.39344    0.09353   4.206 0.000337 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.133 on 23 degrees of freedom
Multiple R-Squared: 0.4348,     Adjusted R-squared: 0.4102
F-statistic: 17.69 on 1 and 23 DF,  p-value: 0.0003366
```

Die Güte des Modells auf Basis Δ-Anzahl-Dateien ist noch ein wenig schlechter als auf Basis zweier Prediktoren und damit ebenfalls nicht akzeptabel.

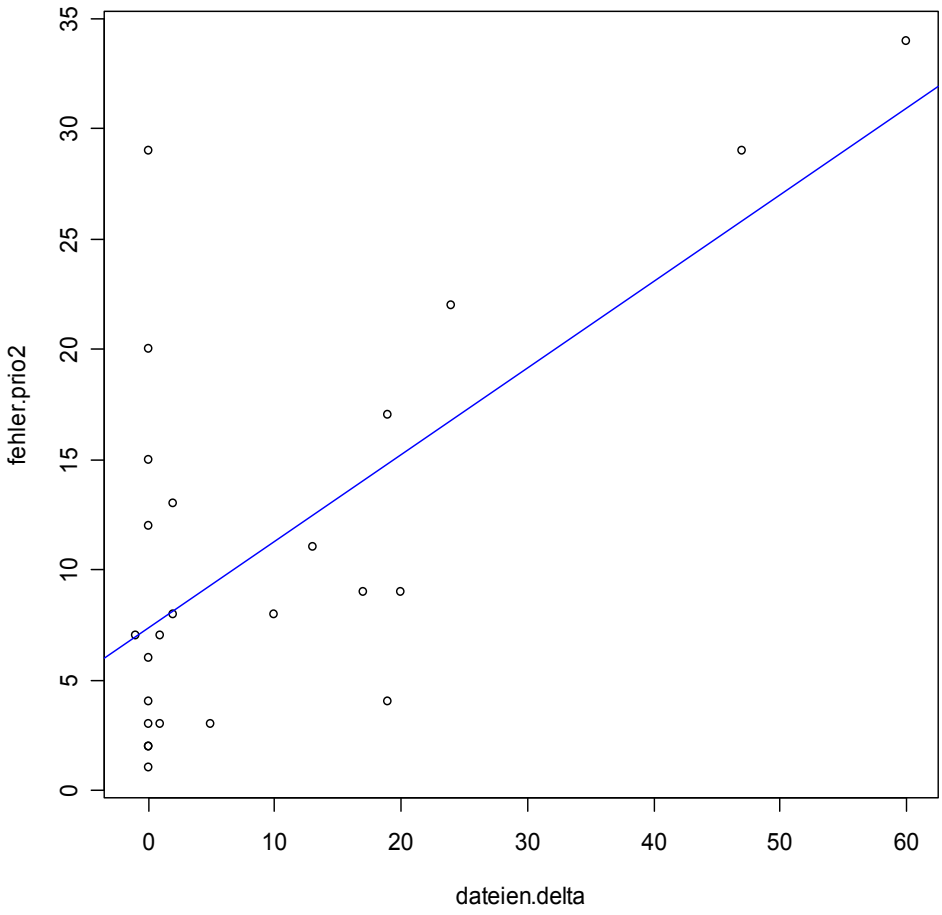

Abbildung 4-30: DOT-Plot Anzahl Feldfehler mit Priorität 2 mit Δ-Anzahl-Dateien

Im DOT-Plot der Abbildung 4-30 liegt der Achsenabschnitt der Regressionsgeraden bei 7,3587 und ergibt sich, wie auch aus dem Diagramm ersichtlich, aus einer relativ breiten Streuung der Anzahl Feldfehler mit Priorität 3 bei Δ-Anzahl-Dateien = 0.

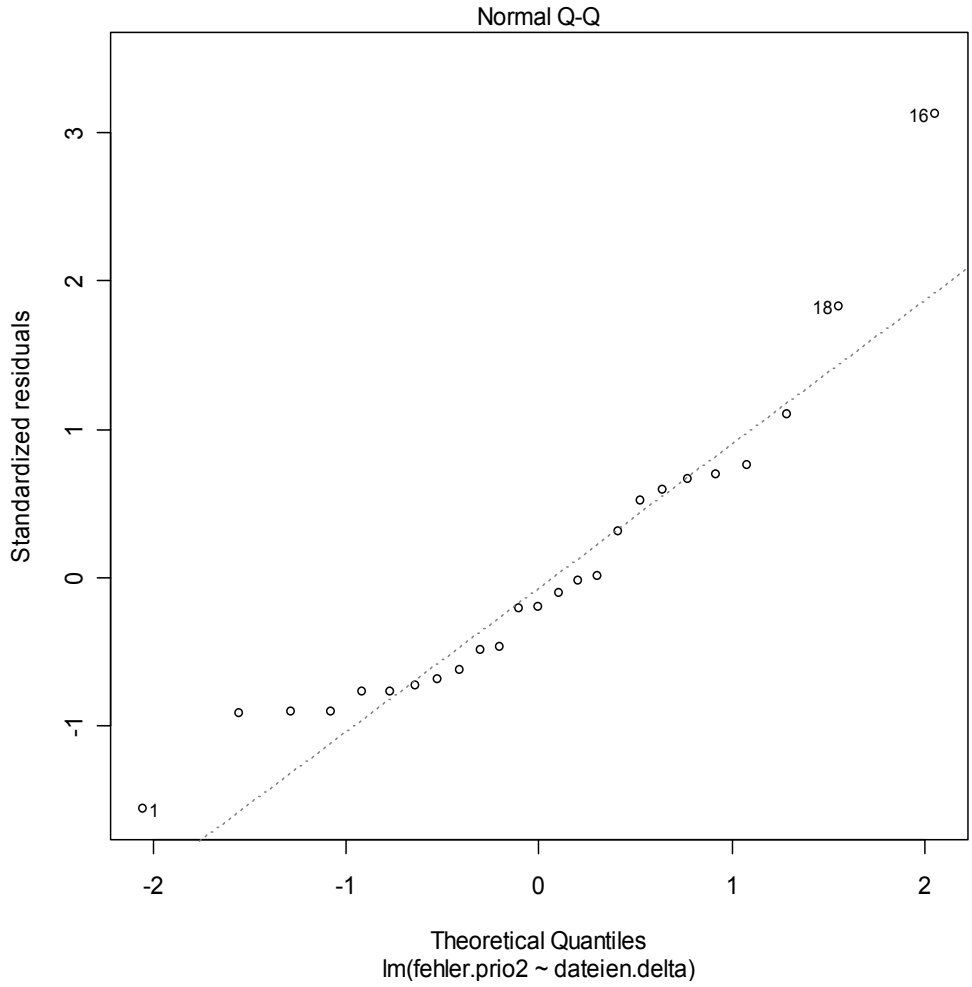

Abbildung 4-31: QQ-Plot Anzahl Feldfehler mit Priorität 2 mit Δ-Anzahl-Dateien

Der QQ-Plot in Abbildung 4-31 mit Δ-Anzahl-Dateien als erklärende Variable zeigt mit Ausnahme von 4 bis 5 Ausreißern zwar eine gute Übereinstimmung der theoretischen Residuen mit der Normalverteilungsannahme der Residuen, ergibt aber insgesamt eine etwas schlechtere Ausgangsbasis als der QQ-Plot der linearen Regression mit LOC und Δ-Anzahl-Dateien als erklärende Variable.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio2
              Df  Sum Sq Mean Sq F value    Pr(>F)
dateien.delta  1  900.32  900.32  17.694 0.0003366 ***
Residuals     23 1170.32   50.88
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt eine hohe Signifikanz von Δ-Anzahl-Dateien zur Erklärung der Anzahl Feldfehler mit Priorität 3, allerdings ist die Signifikanz geringer als bei der

linearen Regression mit LOC und Δ-Anzahl-Dateien als erklärende Variable. Überhaupt ist dieser Signifikanzwert verglichen mit den Signifikanzwerten der linearen Regressionen der Gesamtanzahl Feldfehler sowie der Anzahl Feldfehler mit Priorität 3 und 4 um etwa 2 Zehnerpotenzen geringer. Das könnte auch in diesem Fall die schlechten Bestimmtheitsmaße dieses Modells erklären.

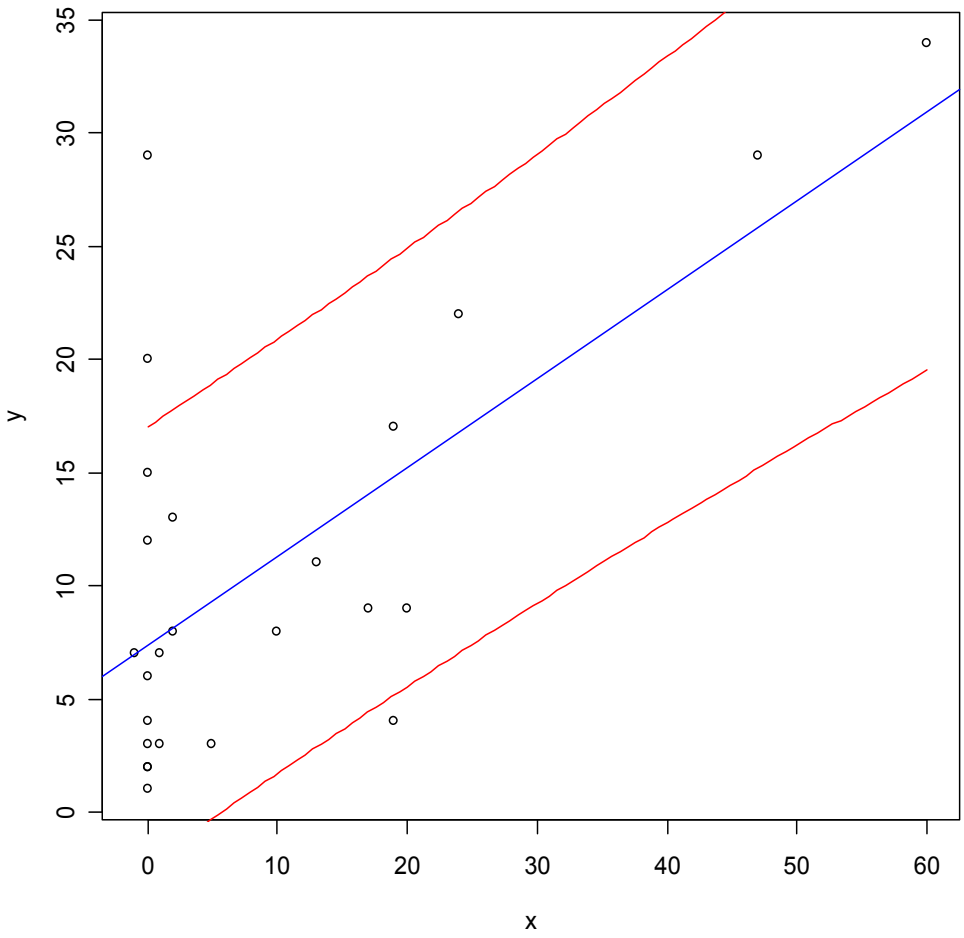

Abbildung 4-32: Prognoseintervall Anzahl Feldfehler mit Priorität 2 mit Δ-Anzahl-Dateien

Auch das Prognoseintervall zum Niveau 80% in Abbildung 4-32 mit einer Schwankungsbreite von etwa ± 10 Fehlern der Priorität 2 bestätigt die bereits vermutete schlechte Qualität dieser linearen Regression. Die Breite des Prognoseintervalls ist damit größer als 50% der betrachteten maximalen Feldfehleranzahl.

Zum Vergleich auch hier wieder die lineare Regression mit Δ-LOC als erklärende Variable.

Lineare Regression mit Δ-LOC:

```
lm(formula = fehler.prio2 ~ loc.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-14.276  -5.578  -1.636   4.230  20.730

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 7.4495854  1.9555867   3.809 0.000902 ***
loc.delta   0.0005304  0.0001664   3.187 0.004104 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.902 on 23 degrees of freedom
Multiple R-Squared: 0.3063,     Adjusted R-squared: 0.2762
F-statistic: 10.16 on 1 and 23 DF,  p-value: 0.004104
```

Dieses Modell ist mit einem Bestimmtheitsmaß von 0,3063 nicht akzeptabel.

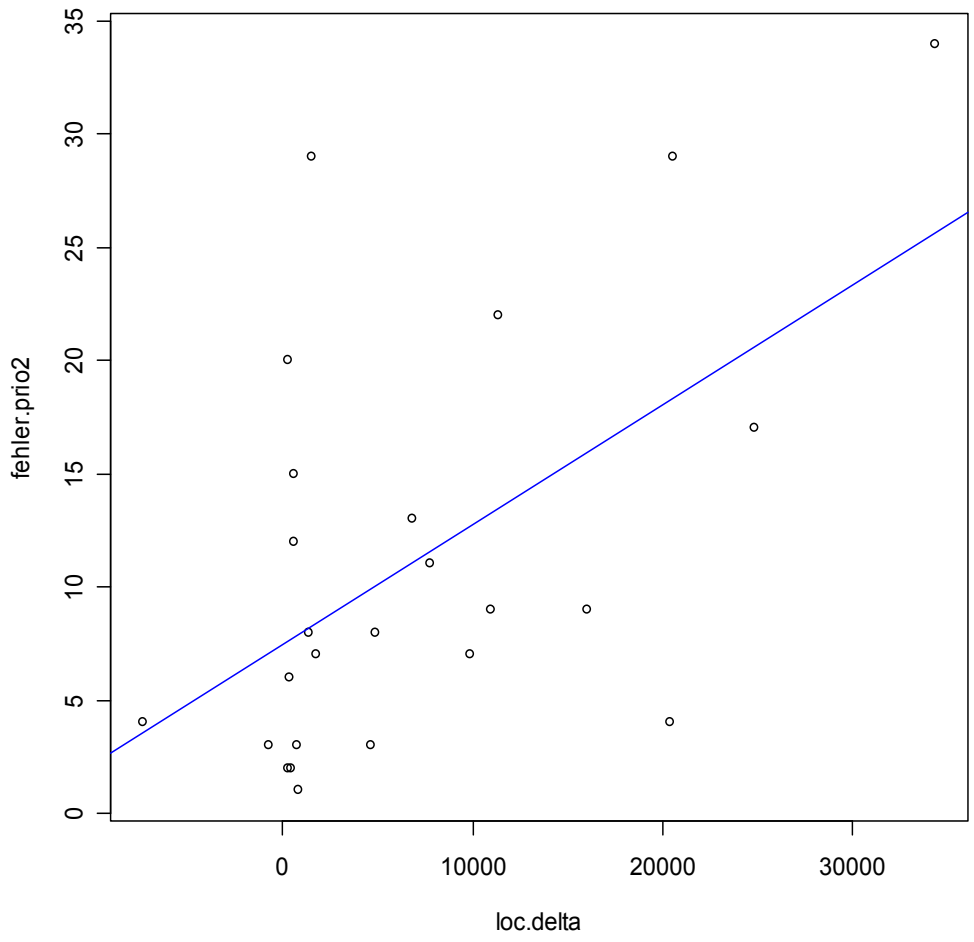

Abbildung 4-33: DOT-Plot Anzahl Feldfehler mit Priorität 2 mit Δ-LOC

Der Achsenabschnitt der Regressionsgeraden im DOT-Plot der Abbildung 4-33 liegt bei 7,45 und damit sehr nahe dem Achsenabschnitt der linearen Regression mit Δ-Anzahl-Dateien. Der Achsenabschnitt ergibt sich auch hier, wie aus dem Diagramm ersichtlich, aus einer relativ breiten Streuung der Anzahl Feldfehler mit Priorität 2 bei Δ-LOC = 0.

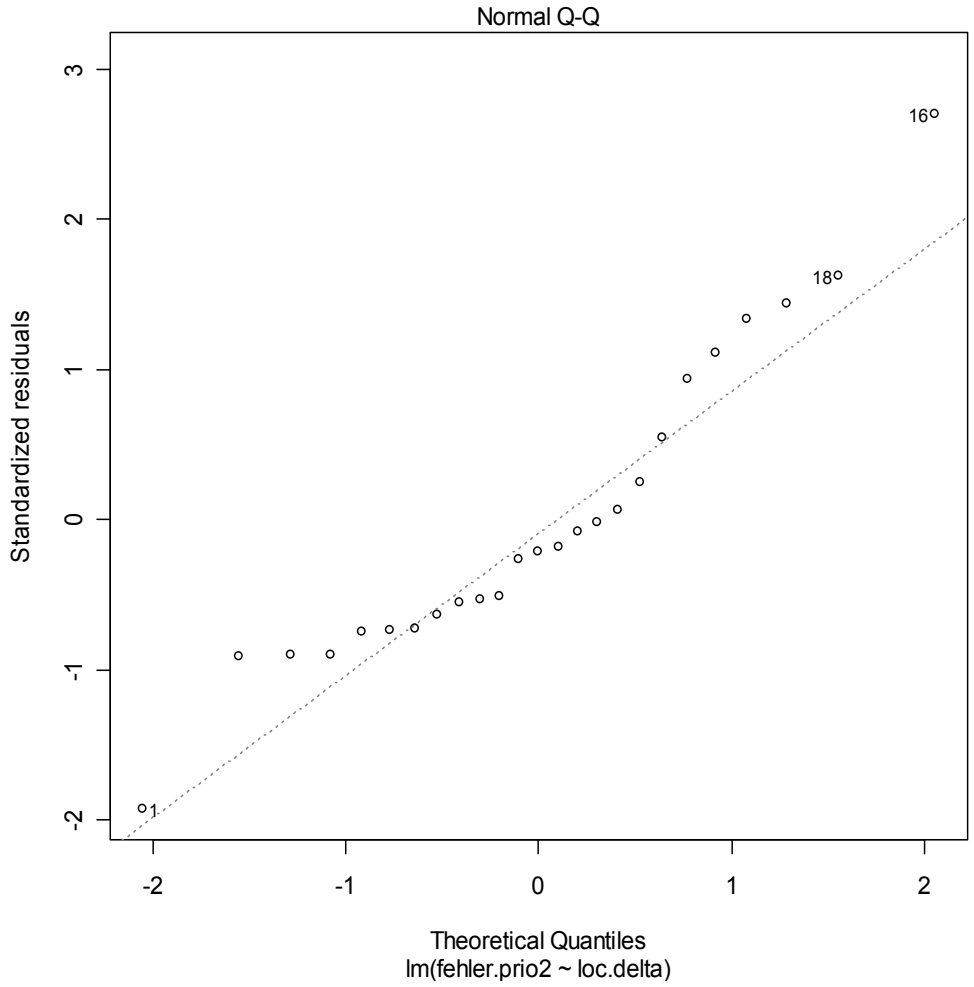

Abbildung 4-34: QQ-Plot Anzahl Feldfehler mit Priorität 2 mit Δ-LOC

Der QQ-Plot in Abbildung 4-34 mit Δ-LOC als erklärende Variable ist ähnlich zu dem QQ-Plot der linearen Regression mit Δ-Anzahl-Dateien als erklärende Variable. Auch in diesem Fall ist die Normalverteilungsannahme der Residuen annähernd erfüllt, wenn auch schlechter als bei der linearen Regression mit LOC und Δ-Anzahl-Dateien als erklärende Variable.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio2
           Df  Sum Sq Mean Sq F value    Pr(>F)
loc.delta   1  634.31  634.31  10.157 0.004104 **
Residuals  23 1436.33   62.45
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt nur mehr eine mittlere Signifikanz von Δ-LOC zur Erklärung der Gesamtfehleranzahl und damit eine um den Faktor 10 geringere Signifikanz als die Signifikanz von Δ-Anzahl-Dateien.

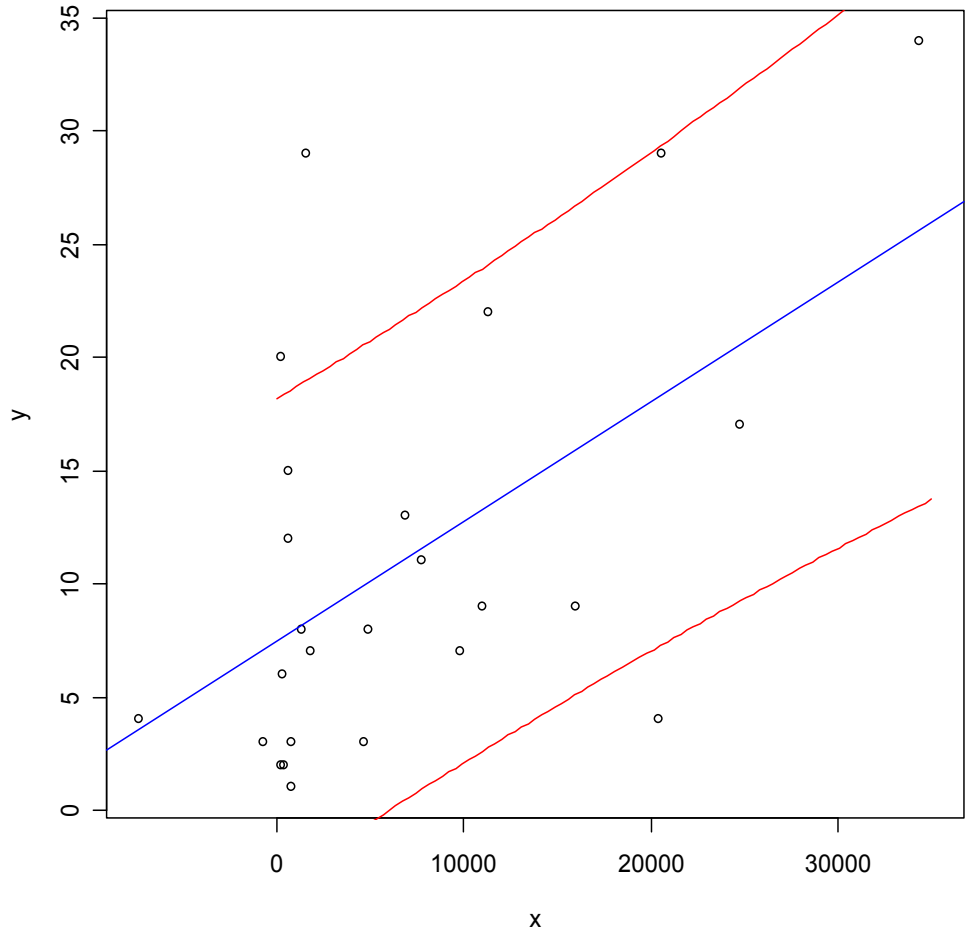

Abbildung 4-35: Prognoseintervall Anzahl Feldfehler mit Priorität 2 mit Δ-LOC

Das Prognoseintervall zum Niveau 80% in Abbildung 4-35 mit einer Schwankungsbreite von etwa ± 15 Fehlern der Priorität 2 bei einer betrachteten Maximalfehleranzahl von 35 zeigt ebenfalls auf, dass diese lineare Regression aller Wahrscheinlichkeit nach keine brauchbaren Ergebnisse liefern wird.

4.6.5 Detailanalyse der Anzahl Feldfehler mit Priorität 1

Die Modellparameter ergeben sich zu:

```
lm(formula = fehler.prio1 ~ loc + dateien.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-10.898  -2.381  -2.061   1.076  14.538

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.097e+00  6.531e+00   0.321    0.751
loc           1.566e-06  3.004e-05   0.052    0.959
dateien.delta 5.719e-01  8.939e-02   6.398 1.95e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.365 on 22 degrees of freedom
Multiple R-Squared: 0.6797,     Adjusted R-squared: 0.6506
F-statistic: 23.35 on 2 and 22 DF,  p-value: 3.636e-06
```

Die Güte dieses Modells ist mit einem Bestimmtheitsmaß von 0.6797 die zweithöchste aller im Rahmen dieser Fallstudie bisher berechneten Werte.

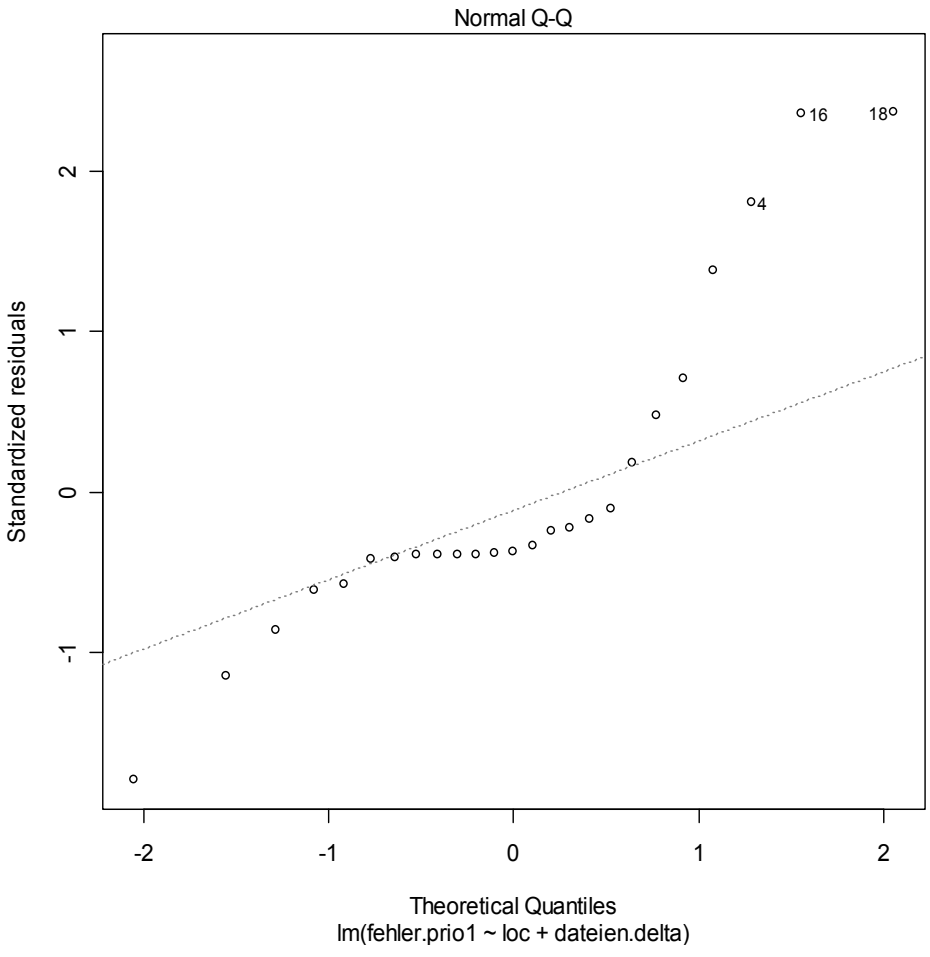

Abbildung 4-36: QQ-Plot Anzahl Feldfehler mit Priorität 1 mit 2 Prediktoren

Im Fall der Feldfehler mit Priorität 1 zeigt der QQ-Plot in Abbildung 4-36 bereits relativ gravierende Abweichungen von der für die lineare Regression notwendigen Normalverteilungsannahme der Residuen. Die Aussage der linearen Regression im Fall der Feldfehler mit Priorität 1 erscheint daher trotz des relativ guten Bestimmtheitsmaßes fragwürdig. Obwohl der QQ-Plot des Modells Anzahl Feldfehler mit Priorität 1 mit allen Prediktoren (Abbildung 4-4) ebenfalls bereits eine fragwürdige Überein-stimmung mit der Normalverteilungsannahme der Residuen zeigt, ist dieses Modell jedenfalls dem Modell mit Δ-Anzahl-Dateien und LOC als Prediktoren vorzuziehen.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio1
              Df  Sum Sq Mean Sq F value    Pr(>F)
loc            1  233.40  233.40  5.7618   0.02528 *
dateien.delta  1 1658.05 1658.05 40.9311 1.946e-06 ***
Residuals     22  891.18   40.51
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die Signifikanz von Δ-Anzahl-Dateien ist hier noch immer sehr hoch, während die Signifikanz von LOC wesentlich schwächer ist. Zur einfacheren graphischen Darstellung bzw. Analyse betrachten wir im Folgenden daher wieder die lineare Regression nur mehr mit der erklärenden Variablen Δ-Anzahl-Dateien.

Lineare Regression mit Δ-Anzahl-Dateien:

```
lm(formula = fehler.prio1 ~ dateien.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-10.833  -2.428  -1.999   1.142  14.572

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.42839    1.46936   1.653    0.112
dateien.delta  0.57025    0.08163   6.986 4.04e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.225 on 23 degrees of freedom
Multiple R-Squared: 0.6797,     Adjusted R-squared: 0.6658
F-statistic: 48.81 on 1 and 23 DF,  p-value: 4.040e-07
```

Die Güte dieses Modells entspricht exakt dem Wert des Modells auf Basis zweier Prediktoren.

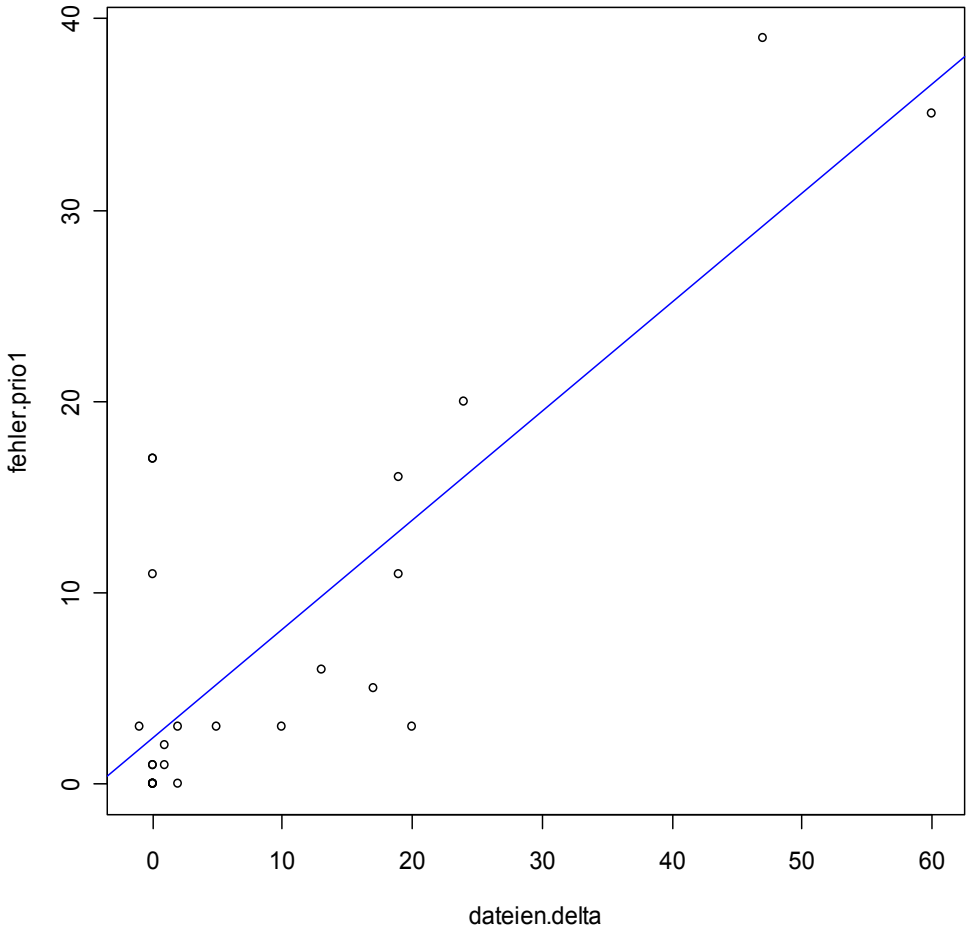

Abbildung 4-37: DOT-Plot Anzahl Feldfehler mit Priorität 1 mit Δ-Anzahl-Dateien

Im DOT-Plot der Abbildung 4-37 liegt der Achsenabschnitt der Regressionsgeraden bei 2,4284 und damit im Verhältnis zur höchsten beobachteten Fehleranzahl sehr nahe beim Nullpunkt. Die Streuung der beobachteten Fehleranzahlen im Bereich Δ-Anzahl-Dateien < 20 ist relativ groß.

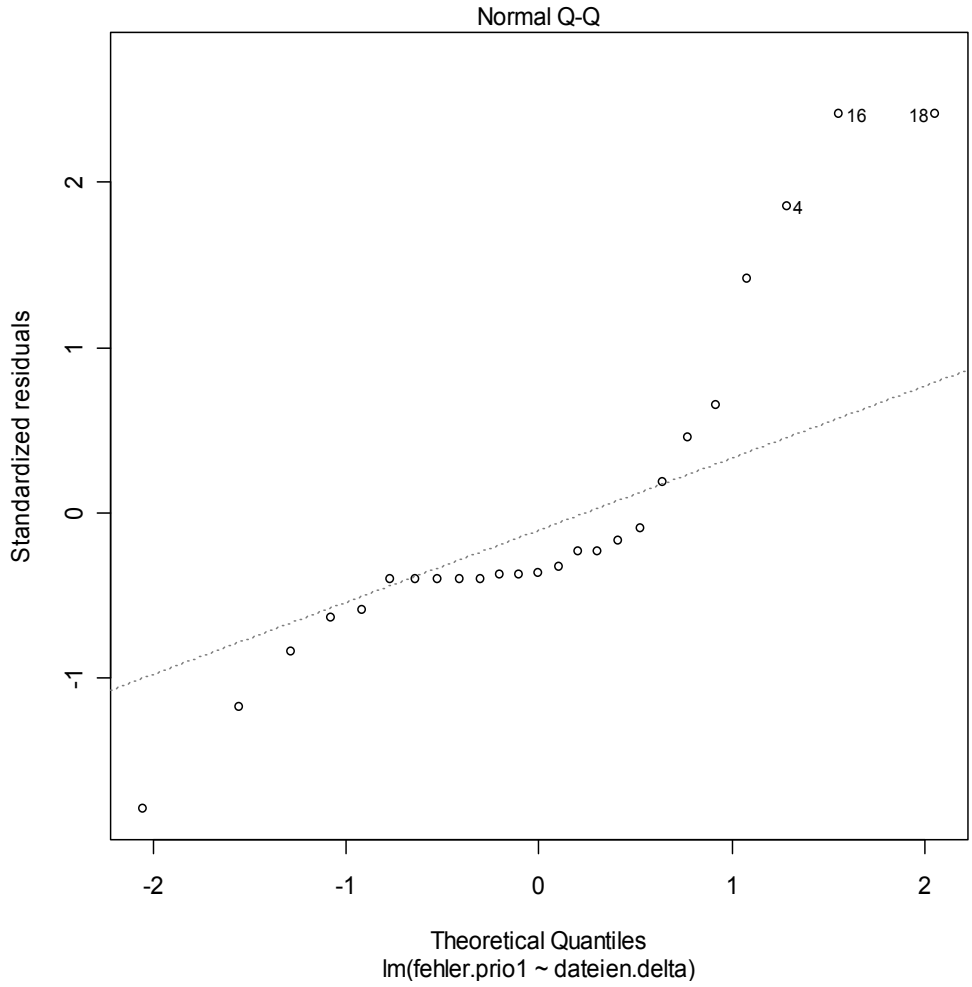

Abbildung 4-38: QQ-Plot Anzahl Feldfehler mit Priorität 1 mit Δ-Anzahl-Dateien

Auch der QQ-Plot in Abbildung 4-38 mit Δ-Anzahl-Dateien als erklärende Variable zeigt bereits relativ gravierende Abweichungen von der für die lineare Regression notwendigen Normalverteilungsannahme der Residuen. Die Aussage der linearen Regression im Fall der Feldfehler mit Priorität 1 erscheint daher auch bei Verwendung von Δ-Anzahl-Dateien als einzige erklärende Variable und trotz des relativ guten Bestimmtheitsmaßes sowie der guten Nullpunkübereinstimmung fragwürdig.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio1
              Df  Sum Sq Mean Sq F value    Pr(>F)
dateien.delta  1 1891.35 1891.35  48.806 4.040e-07 ***
Residuals     23  891.29   38.75
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt trotz der Abweichungen im QQ-Plot eine hohe Signifikanz von Δ-Anzahl-Dateien zur Erklärung der Anzahl Feldfehler mit Priorität 1. Die Signifikanz ist gegenüber der linearen Regression mit LOC und Δ-Anzahl-Dateien als erklärenden Variable sogar noch gestiegen und stimmt mit den guten Signifikanzwerten der linearen Regressionen der Gesamtanzahl Feldfehler (siehe auch Kapitel 4.6.1) sowie der Anzahl Feldfehler mit Priorität 3 und 4 überein (siehe auch Kapitel 4.6.2 und 4.6.3).

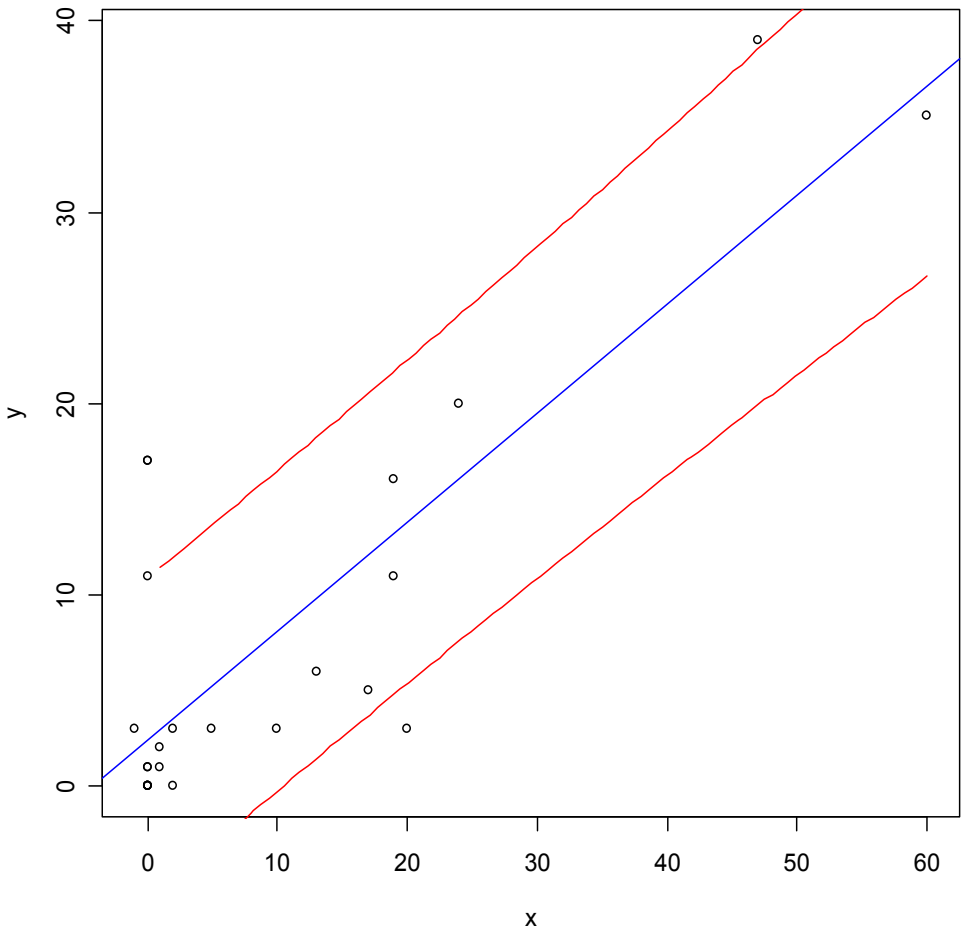

Abbildung 4-39: Prognoseintervall Anzahl Feldfehler mit Priorität 1 mit Δ-Anzahl-Dateien

Interessanterweise ist das Prognoseintervall zum Niveau 80% in Abbildung 4-39 mit einer Schwankungsbreite von etwa ± 5 Feldfehlern der Priorität 1 im Verhältnis zur maximalen Anzahl der beobachteten Feldfehler und im Vergleich zu den vorangegangenen Untersuchungen relativ gut. Offenbar wiegt hier das relativ hohe Bestimmtheitsmaß schwerer als die relativ schlechte Übereinstimmung des QQ-Plots mit der Normalverteilungsannahme der Residuen.

Zum Vergleich auch hier wieder die lineare Regression mit Δ-LOC als erklärende Variable.

Lineare Regression mit Δ-LOC:

```
lm(formula = fehler.prio1 ~ loc.delta)

Residuals:
    Min      1Q  Median      3Q     Max
-11.901  -4.829  -2.541   3.206  20.562

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.517546   1.950390   1.291 0.209601
loc.delta   0.000775   0.000166   4.669 0.000106 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.881 on 23 degrees of freedom
Multiple R-Squared: 0.4866,     Adjusted R-squared: 0.4642
F-statistic:  21.8 on 1 and 23 DF,  p-value: 0.0001063
```

Hier zeigt sich analog zu den anderen Modellen auf Basis ausgesuchter Prediktoren wieder das schlechtere Abschneiden von Δ-LOC als Prediktor. Ein Bestimmtheitsmaß von 0,4866 ist eigentlich nicht mehr akzeptabel.

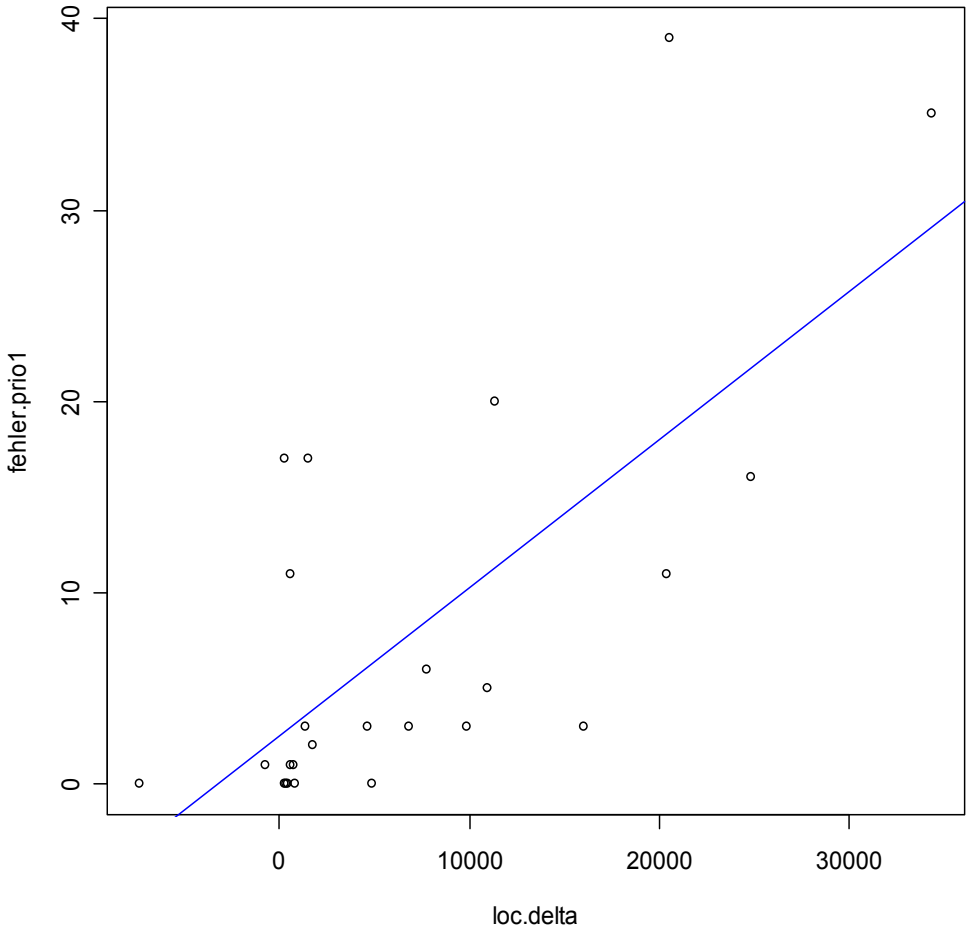

Abbildung 4-40: DOT-Plot Anzahl Feldfehler mit Priorität 1 mit Δ-LOC

Im DOT-Plot der Abbildung 4-40 liegt der Achsenabschnitt der Regressionsgeraden bei 2,5175 und damit sehr nahe dem Achsenabschnitt der linearen Regression mit Δ-Anzahl-Dateien. Die Streuung der beobachteten Fehleranzahlen ist allerdings insgesamt groß.

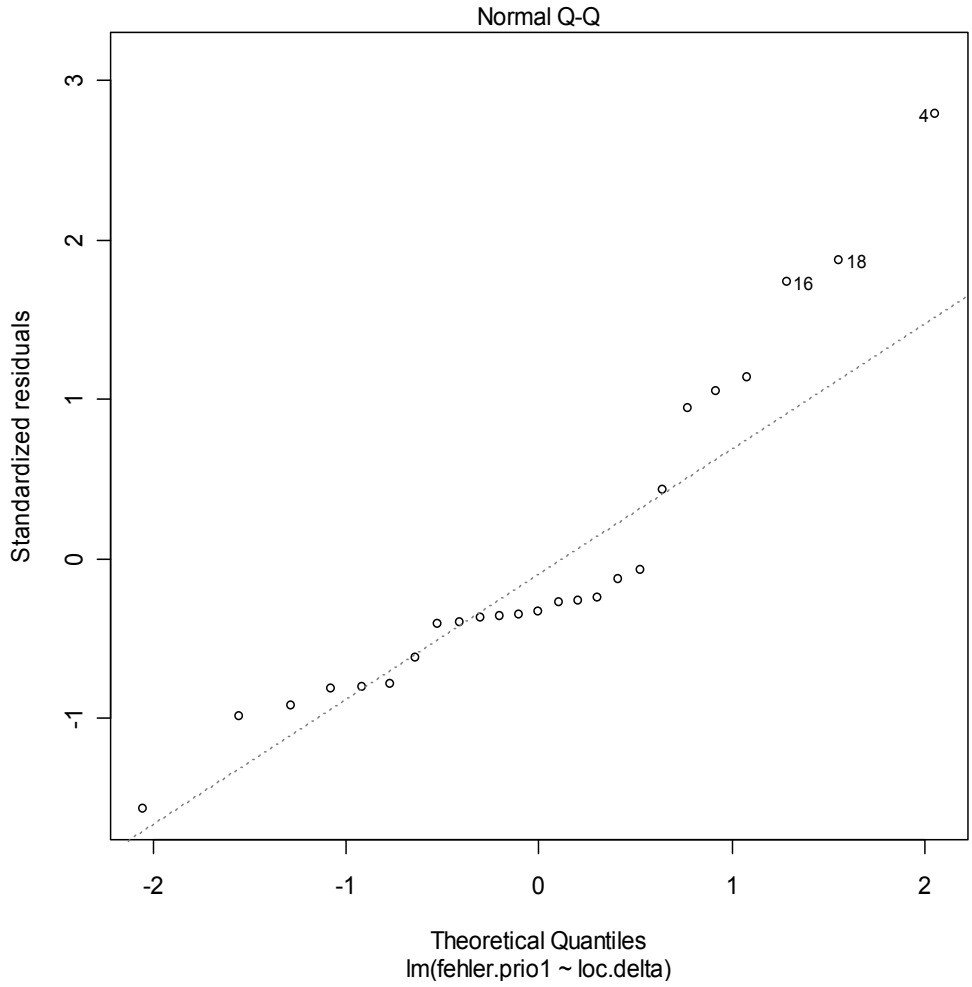

Abbildung 4-41: QQ-Plot Anzahl Feldfehler mit Priorität 1 mit Δ-LOC

Der QQ-Plot in Abbildung 4-41 mit Δ-LOC als erklärende Variable zeigt ein wesentlich besseres Bild als der QQ-Plot der linearen Regression mit Δ-Anzahl-Dateien als erklärende Variable. In diesem Fall ist die Normalverteilungsannahme der Residuen mit Ausnahme eines Ausreißers wieder annähernd erfüllt.

Analysis of Variance (ANOVA):

```
Analysis of Variance Table

Response: fehler.prio1
           Df  Sum Sq Mean Sq F value    Pr(>F)
loc.delta   1 1353.93 1353.93  21.796 0.0001063 ***
Residuals  23 1428.71   62.12
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die ANOVA zeigt auch für Δ-LOC als erklärende Variable eine hohe Signifikanz, allerdings um 3 Zehnerpotenzen geringer als mit Δ-Anzahl-Dateien.

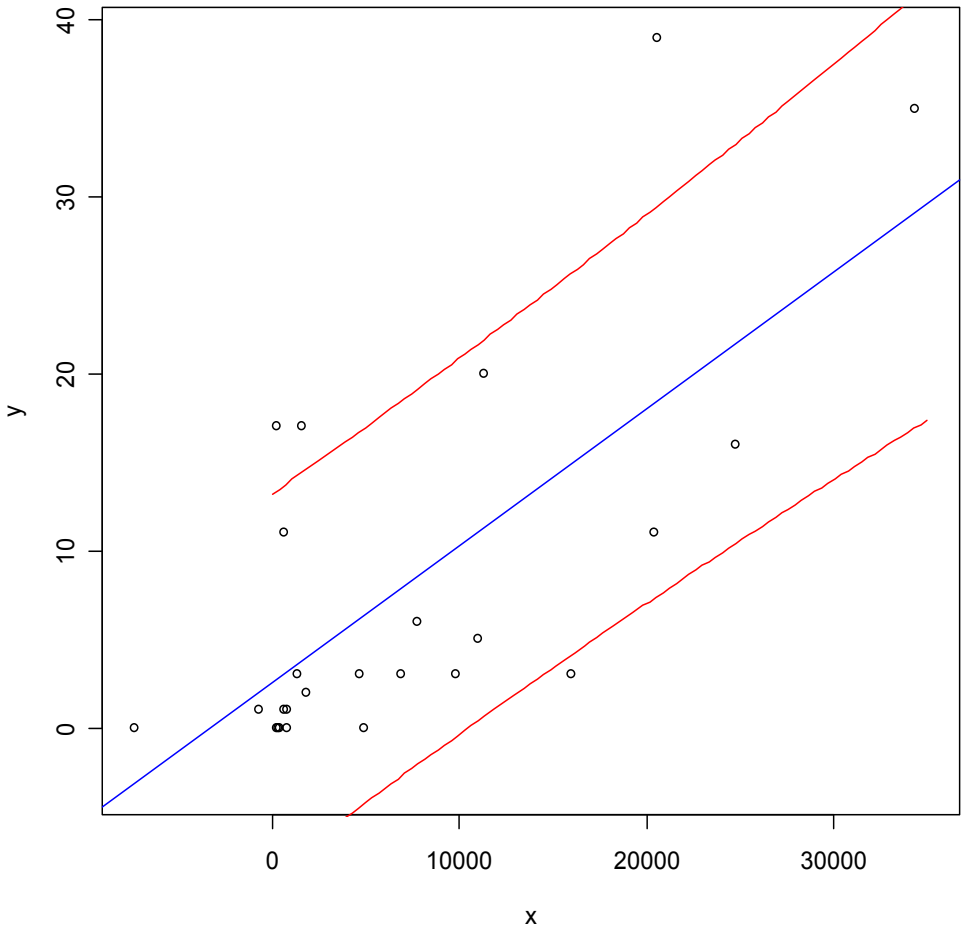

Abbildung 4-42: Prognoseintervall Anzahl Feldfehler mit Priorität 1 mit Δ-LOC

Das Prognoseintervall zum Niveau 80% in Abbildung 4-42 mit einer Schwankungsbreite von etwa ± 10 Feldfehlern der Priorität 1 ist etwa doppelt so breit wie bei der linearen Regression mit Δ-Anzahl-Dateien als erklärende Variable. Darauf haben bereits sowohl die um 3 Zehnerpotenzen geringere Signifikanz im Rahmen der ANOVA als auch die schlechte Modellgüte hingedeutet. Bei einer betrachteten Maximalfehleranzahl von 40 ist eine Schwankungsbreite von ± 10 Feldfehlern im Bereich von 50% der maximalen beobachteten Fehleranzahl. Auch dieses Modell muss daher mit Vorsicht interpretiert werden.

4.7 Prognoserechnung

Für die 4 Testdatensätze der Releases V2.5.08 bis V2.5.11 wurde jeweils eine Prognoserechnung anhand der ermittelten Modellparameter durchgeführt. Im Folgenden werden die Prognoseergebnisse im Detail betrachtet.

Die Tabellen enthalten dabei jeweils in der hellgelb hinterlegten Zeile die echten Werte des Testdatensatzes mit der Bezeichnung der jeweiligen Test-Release. Die Spalten enthalten die Werte für die mittlere Abweichung einer Prognoserechnung über alle Fehlerklassen und dann je Fehlerklasse (Fehler mit Priorität 1 bis 4) die berechnete Anzahl Feldfehler sowie die absoluten und relativen Abweichungen. Fehler ohne Priorität und Fehler der Priorität 0 werden aus den in diesem Kapitel bereits erwähnten Gründen nicht mehr im Detail betrachtet.

Darunter befinden sich die Ergebnisse der Regressionsrechnungen mit allen Prediktoren, LOC + Δ-Anzahl-Dateien, Δ-Anzahl-Dateien sowie Δ-LOC.

4.7.1 Release V2.5.08

Tabelle 4-3: Prognoserechnung V2.5.08

		Mittlere Abw.	Tot.		0	1	2	3	4
	V2.5.08		65	0	0	12	26	20	7
Alle Prediktoren			38,074	0,046	-0,607	1,879	8,698	21,65	5,5814
Abweichung abs.			-26,93	0,046	-0,607	-10,1	-17,3	1,648	-1,419
Abweichung rel.		-41%	-41%			-84%	-67%	8%	-20%
LOC + Δ–Dateien			39,765			2,493	9,026	22,36	6,1933
Abweichung abs.			-25,24			-9,51	-17	2,362	-0,807
Abweichung rel.		-37%	-39%			-79%	-65%	12%	-12%
Δ-Dateien			36,633			2,428	7,359	17,26	6,2903
Abweichung abs.			-28,37			-9,57	-18,6	-2,74	-0,71
Abweichung rel.		-44%	-44%			-80%	-72%	-14%	-10%
Δ-LOC			41,619			3,496	8,119	19,31	7,8289
Abweichung abs.			-23,38			-8,5	-17,9	-0,69	0,8289
Abweichung rel.		-33%	-36%			-71%	-69%	-3%	12%

Regression mit allen Prediktoren:

Die Abweichung bei der Gesamtanzahl aller Feldfehler beträgt absolut -26,93. Der wahre Wert von 65 Feldfehlern wird also um 41% unterschätzt.

Die Prognose der Anzahl Feldfehler mit Priorität 3 und Priorität 4 trifft mit einer absoluten Abweichung von 1,648 bzw. -1,419 die wahren Werte ziemlich genau. Für Feldfehler der Priorität 3 beträgt die relative Abweichung lediglich 8%, während die relative Abweichung für Feldfehler der Priorität 4 mit -20% auf Grund der relativ geringen Gesamtanzahl hoch ausfällt.

Die größten Abweichungen gibt es in diesem Fall bei Feldfehlern der Priorität 1 und 2. Die absolute Abweichung liegt für Feldfehler der Priorität 1 bei -10,1 bzw. bei -17,3 für Feldfehler der Priorität 2. Die relativen Abweichungen in Bezug auf die wahren Fehleranzahlen fallen mit -84% bzw. -67% auch entsprechend hoch aus.

Regression mit ausgewählten Prediktoren:

Ähnlich verhält es sich mit den Regressionsrechnungen mit ausgesuchten Prediktoren. Im Durchschnitt die besten Ergebnisse werden mit dem Prediktor Δ-LOC erzielt, obwohl im Rahmen der Modellanalyse keinen wesentlichen Unterschiede zu den Modellen mit 2 Prediktoren bzw. Δ-Anzahl-Dateien als Prediktor gefunden wurden. Δ-LOC entspricht genauso wie Δ-Anzahl-Dateien annähernd der Entwicklungsmetrik „Änderungen im Code“ nach [Li06]. [Knab06] hat in diesem Zusammenhang auch festgestellt, dass Änderungsmetriken eine große Bedeutung als Prediktor zur Vorhersage der Anzahl Feldfehler haben.

Die mittlere relative Abweichung beträgt mit dem Prediktor Δ-LOC -33%, am besten stimmt der errechnete Prognosewert der Feldfehler mit Priorität 3 mit der Realität überein (absolut -0,69 oder relativ -3%). Aber auch die Prognose für Feldfehler der Priorität 4 ist mit einer absoluten Abweichung von 0,8289 als sehr gut zu bezeichnen. Diese Ergebnisse decken sich auch mit den in den vorangegangenen Kapiteln durchgeführten Modellanalysen.

Die Ergebnisse des Modells mit Δ-Dateien als Prediktor für die Fehlerklassen 1 und 2 sowie des Modells mit Δ-LOC als Prediktor für die Fehlerklasse 2 liegen außerhalb der jeweiligen Prognoseintervalle.

Da die im Rahmen dieser Fallstudie verfügbaren Prediktoren linear voneinander abhängig sind, ist die Prognosequalität der einzelnen Prediktoren untereinander offensichtlich immer vergleichbar. Feldfehler der Priorität 1 und 2 werden im Fall der Release V2.5.08 von jeder Prediktoren-Kombination schlecht prognostiziert. Feldfehler der Priorität 3 und 4 werden mit jeder Prediktoren-Kombination gut prognostiziert.

Zusammenfassung:

Als Entscheidungsbasis für Projektmanager sind primär die Vorhersagen für Feldfehler der Prioritäten 3 und 4 interessant. Im Fall des Testdatensatzes V2.5.08 ergibt sich mit dem Prediktor Δ-LOC dafür eine gut verwendbare Vorhersage mit Abweichungen von -0,69 (Feldfehler der Priorität 3) und 0,8289 (Feldfehler der Priorität 4).

Im Rahmen der Modellanalyse wurde zwar für die Interpretation der Modellergebnisse für Feldfehler der Priorität 4 Vorsicht geboten, im konkreten Fall der Release V2.5.08 ist die Ergebnisqualität jedoch akzeptabel.

4.7.2 Release V2.5.09

Tabelle 4-4: Prognoserechnung V2.5.09

		Mittlere Abw.	Tot.		0	1	2	3	4
V2.5.09			36	2	0	4	14	15	1
Alle Prediktoren			80,333	-0,522	0,417	9,878	14,76	39,77	16,177
Abweichung abs.			44,333	-2,522	0,417	5,878	0,762	24,77	15,177
Abweichung rel.		305%	123%	-126%		147%	5%	165%	1518%
LOC + Δ–Dateien			81,637			9,938	14,97	40,38	16,357
Abweichung abs.			45,637			5,938	0,967	25,38	15,357
Abweichung rel.		397%	127%			148%	7%	169%	1536%
Δ-Dateien			76,946			9,842	12,47	32,74	16,502
Abweichung abs.			40,946			5,842	-1,53	17,74	15,502
Abweichung rel.		383%	114%			146%	-11%	118%	1550%
Δ-LOC			65,311			7,708	11	28,27	13,552
Abweichung abs.			29,311			3,708	-3	13,27	12,552
Abweichung rel.		299%	81%			93%	-21%	88%	1255%

Regression mit allen Prediktoren

Die Abweichung bei der Gesamtanzahl aller Feldfehler beträgt absolut 44,333. Der wahre Wert von 36 Feldfehlern wird also um 123% überschätzt.

Nur die Prognose der Anzahl Feldfehler mit Priorität 2 trifft mit einer absoluten Abweichung von 0,762 den wahren Wert sehr genau. Auch die relative Abweichung von 5% ist sehr gut.

Bei den Fehlerklassen 1, 3 und 4 liegen die absoluten Abweichungen bei 5,878, 24,77 sowie 15,177 Feldfehlern. Die relativen Abweichungen sind aber mit 147%, 165% bzw. 1518% relativ groß. Insbesondere die massive Abweichung bei der als Entscheidungsgrundlage wichtigen Fehlerklasse 4 ist nicht akzeptabel.

Die Modellanalysen der vorangegangenen Kapitel haben zwar gewisse Schwächen für das Modell Anzahl Feldfehler mit Priorität 4 aufgezeigt, dies trifft aber auch auf die Modell Anzahl Feldfehler mit Priorität 1 und 2 zu. Doch gerade die Prognose der Anzahl Feldfehler mit Priorität 2 liefert in diesem Fall das beste Ergebnis.

Regression mit ausgesuchten Prediktoren

Wieder ist die mittlere relative Abweichung des Prediktors Δ-LOC im Vergleich mit den anderen Regressionsrechnungen am geringsten, mit 299% insgesamt aber trotzdem sehr hoch. Wie auch schon bei der Regression mit allen Prediktoren, liefert die Prognose der Anzahl Feldfehler mit Priorität 2 interessanterweise das beste Ergebnis.

Betrachtet man die Ergebnisse für Feldfehler der Priorität 2 aber im Detail, kann man feststellen, dass die beste Vorhersage aus der Regressionsrechnung mit allen Prediktoren stammt. Dort beträgt die absolute Abweichung 0,762 Feldfehler. Bei der Regressionsrechnung mit dem Prediktor Δ-LOC beträgt die Abweichung absolut -3 Feldfehler und relativ immerhin -21%. Dieses Ergebnis wird durch die Modellanalysen der vorangegangenen Kapitel aber wieder bestätigt.

Die Ergebnisse des Modells mit Δ-Anzahl-Dateien als Prediktor für die Fehlerklasse 4 sowie des Modells mit Δ-LOC als Prediktor für die Fehlerklasse 4 liegen außerhalb des jeweiligen Prognoseintervalls.

Zusammenfassung:

Im Bereich der für einen Projektmanager besonders interessanten Fehlerklassen 3 und 4 ist die Qualität der Prognosen im Fall der Release V2.5.09 nicht akzeptabel. Problematisch ist in diesem Zusammenhang, dass ein Projektmanager auf Grund der Modellanalysen die sicher nicht akzeptablen Werte für Anzahl Feldfehler der Priorität 1 und 4 vielleicht anzweifeln würde, die ebenfalls schlechten Vorhersagen für Feldfehler mit Priorität 3 allerdings akzeptieren. Dass gerade die Ergebnisse für Feldfehler der Priorität 2 so gut ausfallen ist aus Sicht der Modellanalyse ebenfalls nicht zu erwarten.

Die Veränderungen (Δ-Anzahl-Dateien bzw. Δ-LOC) von Release V2.5.08 zu V2.5.09 sind aber im Vergleich zu den anderen Testdatensätzen am größten. Die meist massive Überschätzung der Feldfehleranzahlen durch das lineare Regressionsmodell ist in diesem Fall wohl darauf zurückzuführen. Warum die Anzahl der Feldfehler im Fall der Release V2.5.09 nicht im Verhältnis zu den im Vergleich höheren Werten von Δ-Anzahl-Dateien bzw. Δ-LOC gestiegen ist, liegt offensichtlich an Gründen, die durch die zur Verfügung stehenden Größenmetriken nicht ausreichend berücksichtigt werden.

4.7.3 Release V2.5.10

Tabelle 4-5: Prognoserechnung V2.5.10

		Mittlere Abw.	Tot.		0	1	2	3	4
V2.5.10			59	4	0	1	10	39	5
Alle Prediktoren			44,986	-0,922	-0,627	2,71	10,06	25,52	7,3084
Abweichung abs.			-14,01	-4,922	-0,627	1,71	0,06	-13,5	2,3084
Abweichung rel.		6%	-24%	-123%		171%	1%	-35%	46%
LOC + Δ–Dateien			43,56			3,078	9,788	24,68	6,9575
Abweichung abs.			-15,44			2,078	-0,21	-14,3	1,9575
Abweichung rel.		36%	-26%			208%	-2%	-37%	39%
Δ-Dateien			39,734			2,999	7,752	18,45	7,0758
Abweichung abs.			-19,27			1,999	-2,25	-20,6	2,0758
Abweichung rel.		27%	-33%			200%	-22%	-53%	42%
Δ-LOC			42,269			3,611	8,198	19,56	7,9858
Abweichung abs.			-16,73			2,611	-1,8	-19,4	2,9858
Abweichung rel.		45%	-28%			261%	-18%	-50%	60%

Regression mit allen Prediktoren:

Die Abweichung bei der Gesamtanzahl aller Feldfehler beträgt absolut -14,01. Der wahre Wert von 59 Feldfehlern wird also um 24% unterschätzt. Die mittlere relative Abweichung der Regressionsrechnung mit allen Prediktoren liegt sogar bei sehr guten 6%. Dies liegt vor allem an den ebenfalls sehr guten Prognosewerten für Feldfehler der Fehlerklassen 2 bis 4.

Vor allem die Anzahl Feldfehler der Priorität 2 wird mit einer absoluten Abweichung von 0,06 Feldfehlern bzw. relativ 1% sehr gut vorhergesagt. Aber auch die Fehlerklassen 3 und 4 werden mit einer Abweichung von -13,5 bzw. 2,3084 (relativ -35% bzw. 46%) noch ausreichend gut prognostiziert.

Einzig die Ergebnisse für Fehlerklassen der Priorität 1 fallen schlecht aus. Bis auf das gute Abschneiden der Prognose Anzahl Feldfehler mit Priorität 2 war dieses Ergebnis auf Grund der in den vorangegangenen Kapiteln durchgeführten Modellanalysen zu erwarten.

Regression mit ausgewählten Prediktoren:

Bei den Regressionsrechnungen mit ausgewählten Prediktoren bringt diesmal der Prediktor Δ-Anzahl-Dateien die besten Ergebnisse, Δ-LOC schneidet am schlechtesten ab.

Auf Grund der linearen Abhängigkeit der Prediktoren finden sich auch hier die besten Ergebnisse im Bereich der Anzahl Feldfehler mit Priorität 2. Bei Feldfehlern dieser Fehlerklasse liefert die Regressionsrechnung mit den Prediktoren Δ-Anzahl-Dateien und Δ-LOC mit absolut -0,21 Feldfehlern bzw. -2% relativer Abweichung

das beste Ergebnis. Auch dies hat sich auf Grund der Modellanalysen erwarten lassen.

Die Ergebnisse für Feldfehler der Priorität 1 sind in Wirklichkeit besser, als sie auf den ersten Blick wirken. Die relativen Abweichungen sind zwar sehr groß, die absoluten Abweichungen aber als Entscheidungsgrundlage für Projektmanager brauchbar.

Das Ergebnis des Modells mit Δ-Anzahl-Dateien als Prediktor für die Fehlerklasse 3 liegt außerhalb des zugehörigen Prognoseintervalls.

Zusammenfassung:

Die Prognoserechnung für die Release V2.5.10 liefert vor allem im wichtigen Bereich der Feldfehler mit Priorität 3 und 4 für einen Projektmanager wieder durchaus akzeptable und mit Ausnahme der Anzahl Feldfehler mit Priorität 2 auch zu erwartende Ergebnisse.

4.7.4 Release V2.5.11

Tabelle 4-6: Prognoserechnung V2.5.11

		Mittlere Abw.	Tot.		0	1	2	3	4
V2.5.11			36	5	0	2	8	21	0
Alle Prediktoren			42,523	-0,186	-1,519	1,895	9,601	24,64	6,1139
Abweichung abs.			6,5235	-5,186	-1,519	-0,1	1,601	3,636	6,1139
Abweichung rel.		-11%	18%	-104%		-5%	20%	17%	
LOC + Δ–Dateien			43,769			3,082	9,899	25,02	6,951
Abweichung abs.			7,7687			1,082	1,899	4,024	6,951
Abweichung rel.		30%	22%			54%	24%	19%	
Δ-Dateien			39,734			2,999	7,752	18,45	7,0758
Abweichung abs.			3,734			0,999	-0,25	-2,55	7,0758
Abweichung rel.		11%	10%			50%	-3%	-12%	
Δ-LOC			48,162			4,659	8,915	21,78	9,4094
Abweichung abs.			12,162			2,659	0,915	0,785	9,4094
Abweichung rel.		45%	34%			133%	11%	4%	

Regression mit allen Prediktoren:

Die Abweichung bei der Gesamtanzahl aller Feldfehler beträgt absolut 6,5235. Der wahre Wert von 36 Feldfehlern wird also um 18% überschätzt. Die mittlere relative Abweichung der Regressionsrechnung mit allen Prediktoren liegt bei sehr guten -11%. Dies liegt in diesem Fall an den sehr guten Prognosewerten für Feldfehler der Fehlerklassen 1 bis 3.

Da in der Release V2.5.11 offenbar keine Feldfehler der Priorität 4 enthalten sind, ist die Prognose für die Anzahl Feldfehler mit Priorität 4 auf Grund der Eigenschaften

des linearen Regressionsmodells klarerweise schlecht. Die relative Abweichung kann in diesem Fall gar nicht angegeben werden.

Die Anzahl Feldfehler der Priorität 1 wird mit einer absoluten Abweichung von -0,1 Feldfehlern bzw. relativ -5% sehr gut vorhergesagt. Aber auch die Fehlerklassen 2 und 3 werden mit einer Abweichung von 1,601 bzw. 3,636 (relativ 20% bzw. 17%) ausreichend gut prognostiziert.

Regression mit ausgewählten Prediktoren:

Bei den Regressionsrechnungen mit ausgewählten Prediktoren bringt auch hier der Prediktor Δ-Anzahl-Dateien die besten Ergebnisse, Δ-LOC schneidet wieder am schlechtesten ab.

Hier finden sich die besten Ergebnisse für die Anzahl Feldfehler vor allem bei den Fehlerklassen 2 und 3. Bei Feldfehlern dieser Fehlerklassen liefern die Regressionsrechnungen mit den Prediktoren Δ-Anzahl-Dateien bzw. Δ-LOC die besten Ergebnisse.

Auch die Ergebnisse für Feldfehler der Priorität 1 sind zumindest in absoluten Zahlen als Entscheidungsgrundlage für Projektmanager verwendbar.

Alle Ergebnisse liegen innerhalb der jeweiligen Prognoseintervalle.

Zusammenfassung:

Die Prognoserechnung für die Release V2.5.11 liefert insgesamt gute Ergebnisse. Im wichtigen Bereich der Feldfehler mit Priorität 3 und 4 sind die Ergebnisse für die Fehlerklasse 3 verwendbar, die Anzahl Feldfehler der Fehlerklasse 4 wird aber massiv überschätzt.

Das Problem bezüglich der Prognose der Anzahl Feldfehler mit Priorität 4 liegt, wie schon erwähnt, in der Natur des linearen Regressionsmodells begründet. Alleine aus der Bewertung der verfügbaren Prediktoren kann dieses Modell nicht auf das korrekte Ergebnis schließen. Offenbar sind auch dafür wieder andere Umstände bzw. Prediktoren verantwortlich, die dem Modell in diesem Fall nicht zur Verfügung stehen.

4.8 Bewertung

Zusammenfassend kann man feststellen, dass die Modelle auf Basis aller Prediktoren die ausgewogensten Ergebnisse liefern. Außer im Fall Anzahl Feldfehler mit Priorität 4, sind diese Modelle in Hinblick auf die Übereinstimmung mit der Normalverteilungsannahme der Residuen immer besser als die Modelle mit einer

Auswahl an Prediktoren. Im Falle der Anzahl Feldfehler mit Priorität 4 ist das Modell mit allen Prediktoren hinsichtlich der Normalverteilungsannahme zumindest gleich gut wie die Modelle mit einer Auswahl an Prediktoren. Die Modellgüte ist bei den Modellen auf Basis aller Prediktoren immer besser als die der Modelle mit ausgesuchten Prediktoren (vergleiche Tabelle 4-7). Auch [Sher05] hat festgestellt, dass die Kombination von Prediktoren oft signifikanter ist als einzelne Prediktoren.

Tabelle 4-7: Bestimmtheitsmaße aller Modelle

	Alle Prediktoren	**Δ-Anzahl-Dateien Δ-LOC**	**Δ-Anzahl-Dateien**	**Δ-LOC**
Alle Feldfehler	0,6414	0,6181	0,6153	0,4711
Prio 1	0,6876	0,6797	0,6797	0,4866
Prio 2	0,5065	0,4699	0,4348	0,3063
Prio 3	0,6118	0,5768	0,5328	0,3961
Prio 4	0,635	0,6004	0,6004	0,4183

Grundsätzlich kann man feststellen, dass die Modellgüte der Modelle auf Basis des Prediktors Δ-LOC erstaunlicherweise immer deutlich schlechter ausfällt als bei allen anderen Modellen.

Im Fall Gesamtanzahl aller Feldfehler sind die Modelle mit ausgewählten Prediktoren hinsichtlich der Normalverteilungsannahme gleich gut. Die zugehörigen Prognoseintervalle zum Niveau 80% sind mit einer möglichen Abweichung von +/- 50 Feldfehlern bei einer Obergrenze von 200 Feldfehlern relativ ungenau (50% der betrachteten Obergrenze).

Im Fall Anzahl Feldfehler mit Priorität 4 sind die Modelle mit 2 Prediktoren und mit dem Prediktor Δ-Anzahl-Dateien hinsichtlich der Normalverteilungsannahme gleich gut. Das Modell mit Δ-LOC als Prediktor schneidet ein wenig besser ab. Insgesamt ist aber die Übereinstimmung mit der Normalverteilungsannahme der Residuen nicht optimal gegeben. Die Bestimmtheitsmaße sind ausgenommen des Modells auf Basis von Δ-LOC im Rahmen der erzielten Ergebnisse akzeptabel. Die zugehörigen Prognoseintervalle zum Niveau 80% sind mit einer möglichen Abweichung von +/- 10 Feldfehlern bei einer Obergrenze von 60 Feldfehlern besser als die des Modells Gesamtanzahl aller Feldfehler (33% der betrachteten Obergrenze).

Im Fall Anzahl Feldfehler mit Priorität 3 sind die Modelle mit ausgewählten Prediktoren hinsichtlich der Normalverteilungsannahme wieder gleich gut. Die Bestimmt-heitsmaße sind wieder mit Ausnahme des Modells auf Basis Δ-LOC gerade noch akzeptabel. Die zugehörigen Prognoseintervalle zum Niveau 80% sind mit einer möglichen Abweichung von +/- 20 Feldfehlern bei einer Obergrenze von 100 Feldfehlern noch akzeptabel (40% der betrachteten Obergrenze).

Im Fall Anzahl Feldfehler mit Priorität 2 ist das Modell mit 2 Prediktoren besser als die Modelle mit Δ-Anzahl-Dateien bzw. Δ-LOC als Prediktor. Die ANOVA liefert bei beiden letztgenannten Modellen einen im Vergleich deutlich schlechteren Signifikanzwert für den jeweiligen Prediktor. Das Prognoseintervall zum Niveau 80% für den Prediktor Δ-Anzahl-Dateien ist mit einer möglichen Abweichung von +/- 10 Feldfehlern bei einer Obergrenze von 35 Feldfehlern nicht akzeptabel (57% der betrachteten Obergrenze). Das Prognoseintervall zum Niveau 80% für den Prediktor Δ-LOC ist mit einer möglichen Abweichung von +/- 15 Feldfehlern bei einer Obergrenze von 35 Feldfehlern noch wesentlich schlechter (86% der betrachteten Obergrenze). Auch die schlechten Bestimmtheitsmaße bestätigen dieses Ergebnis.

Im Fall Anzahl Feldfehler mit Priorität 1 sind die Modelle mit 2 Prediktoren und mit dem Prediktor Δ-Anzahl-Dateien hinsichtlich der Normalverteilungsannahme analog zum Fall Anzahl Feldfehler mit Priorität 4 gleich gut. Das Modell mit Δ-LOC als Prediktor schneidet wieder ein wenig besser ab (obwohl es im Vergleich schlechtere Signifikanzwerte für den zugehörigen Prediktor aufweist). Das Prognoseintervall zum Niveau 80% für den Prediktor Δ-Anzahl-Dateien ist mit einer möglichen Abweichung von +/- 5 Feldfehlern bei einer Obergrenze von 40 Feldfehlern das genaueste der gesamten Fallstudie (25% der betrachteten Obergrenze). Das Prognoseintervall zum Niveau 80% für den Prediktor Δ-LOC ist mit einer möglichen Abweichung von +/- 10 Feldfehlern bei einer Obergrenze von 40 Feldfehlern schon wieder deutlich schlechter (50% der betrachteten Obergrenze).

Teilweise lassen sich die Ergebnisse also mit den Modellanalysen erklären. Im Fall der Release V2.5.09 kommt es jedoch zu in diesem Ausmaß nicht zu erwartenden Abweichungen. Auch die Ergebnisse der Releases V2.5.10 und V2.5.11 decken sich zumindest in Teilbereichen (Anzahl Feldfehler der Priorität 2) nicht mit den aus der Modellanalyse zu erwartenden Ergebnissen. Auf Grund der Modellanalysen war auch nicht zu erwarten, dass in zwei von vier Fällen (V2.5.08 und V2.5.09) das Modell auf Basis Δ-LOC im Mittel die besten Ergebnisse liefert. In den anderen beiden Fällen liefert dieses Modell aber das schlechteste Ergebnis.

Es ist daher nicht möglich, ohne Kenntnis der wahren Werte eine allgemein haltbare Aussage über die Qualität einer Prognoserechnung im Rahmen der Fallstudie zu treffen. Für einen Einsatz in einem realen Projekt sind diese Prognoserechnungen daher nicht geeignet.

Ein wesentliches Problem dieser Fallstudie ist eben die eingeschränkte Verfügbarkeit von Prediktoren. Wie im Kapitel Grundlagen der Feldfehlerprognose von [Li06], aber auch [Knab06] und [Sher05] bereits festgestellt wurde, ist eine breite Palette von voneinander unabhängigen Prediktoren eine ganz wesentliche Grundlage zur Modellierung von erfolgreichen Fehlerprognosemodellen. Es liegt auch die Vermutung nahe, dass sich die auf Basis der Modellanalyse nicht erklärbaren Abweichungen der Ergebnisse dieser Fallstudie wegen der eingeschränkten Prediktorenauswahl ergeben.

Durch die ausschließliche Verwendung von Größenmetriken können viele Aspekte der Software-Entwicklung, die die Menge an Feldfehlern beeinflussen, nicht berücksichtigt werden. Hypothese 1 dieser Arbeit, dass die Qualität von Feldfehlerprognosen von der Wahl der geeigneten Prediktoren abhängt, kann damit eindeutig bestätigt werden.

Ein Grund für die eingeschränkte Prediktorenauswahl liegt auch in der Tatsache, dass sich bestimmte Entwicklungs- und Verwendungsmetriken nach [Li06] (z.B. Änderungen im Code, Anzahl beteiligter Personen, Zeitraum seit Veröffentlichung einer bestimmten Software-Version) nachträglich oft nicht gewinnen lassen. Dies ist auch ganz klar ein Problem dieser Fallstudie. Damit kann auch Hypothese 2 dieser Arbeit, dass nachträgliche Datengewinnung durch eingeschränkte Verfügbarkeit von geeigneten Prediktoren zu Feldfehlerprognosen schlechter Qualität führt, ebenfalls eindeutig bestätigt werden.

Kapitel 5

5 Anwendungsempfehlungen

Damit Feldfehlerprognosen im realen Projektleben Einzug halten können, müssen sie zuverlässig und einfach anzuwenden sein.

Bezüglich Zuverlässigkeit lassen sich aus dieser Arbeit folgende Parameter ableiten:

- Auswahl eines geeigneten Modells[14]
- Verfügbarkeit einer breiten Palette von unabhängigen Metriken bzw. Prediktoren[15]
- Verfügbarkeit von möglichst umfangreichen historischen Daten (Metriken und zugehörigen Fehlerdaten)[16]

Wie schon im Kapitel 1.1 vermutet, gibt es kein universelles Modell für jeden Anwendungsfall in der Feldfehlerprognose. Ein passendes Modell muss für den jeweiligen Einsatzbereich (Prognose der Anzahl Feldfehler, der Feldfehlerdichte, der Feldfehlerrate) sowie das Software-Entwicklungsumfeld (Technologie, Prozess, Personen) erst gefunden werden. Diese Modellfindung kann durchaus aufwendig sein, da es sich um einen empirischen Vorgang handelt. Um ein geeignetes Modell zu finden, muss auch bereits eine gewisse Menge an Basisdaten (Prediktoren, Fehlerdaten) verfügbar sein.

---

[14] Vergleiche [Li06], [Li04], [Sany92]

[15] Vergleiche [Li06], [Knab06]

[16] Vergleiche [Li04]

Dass kein universelles Modell für die Feldfehlerprognose angegeben werden kann, macht die Verwendung allerdings komplex und verhindert derzeit sicher die verbreitete Anwendung von Feldfehlerprognose in realen Projektumfeldern.

Geeignete Metriken (z.B. nach [Li06] Produktmetriken, Entwicklungsmetriken, Verwendungsmetriken, Konfigurationsmetriken) lassen sich grundsätzlich immer aus einem Entwicklungsprozess heraus gewinnen, vorausgesetzt die Metriken werden aufgezeichnet. Das bedingt, dass man schon zu Beginn des Entwicklungsprozesses die gewünschten Metriken und deren Aufzeichnung festlegen sollte. Zur Gewinnung der Produktmetriken sind üblicherweise auch entsprechende Software-Werkzeuge notwendig.

Gerade der Einsatz von Software-Werkzeugen zur Gewinnung verschiedener Produktmetriken kann aber hinsichtlich der möglichen Lizenzgebühren einen Kostenfaktor darstellen, der dem Einsatz in einem realen Projektumfeld ebenfalls entgegensteht.

Wie auch im Bereich der Aufwandschätzung von Software-Systemen ist für eine verlässliche Feldfehlerprognose der Aufbau von historischen Datenbanken notwendig. Wenn die gewünschten Metriken sowie die gewünschten Fehlerdaten definiert sind, ist der Aufbau einer Datenbank mit historischen Metriken bzw. Fehlerdaten grundsätzlich kein Problem. Eigentlich sollten in jedem Projekt- bzw. Produktentwicklungsumfeld Datenbanken mit Metriken und zugehörigen Klassenattributen (wie z.B. Fehlerdaten, Aufwand, etc.) zur späteren Auswertung gepflegt werden.

Hypothese 3 dieser Arbeit, dass die Rahmenbedingungen für die Feldfehlerprognose bereits zu Beginn einer Software-Entwicklung festgelegt werden müssen, um Feldfehlerprognosen hoher Qualität zu erhalten, kann daher auch nur eindeutig bestätigt werden.

Das Potential der Feldfehlerprognose zur Qualitätssteigerung bei gleichzeitigem effizientem Einsatz der Entwicklungsressourcen ist groß. Es lohnt sich daher, weitere Anstrengungen in die Erhöhung der Zuverlässigkeit sowie in die Vereinfachung der Anwendung zu investieren.

Kapitel 6

6 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurden folgende Hypothesen untersucht:

1. Die Qualität von Fehlerprognosen hängt von der Wahl der geeigneten Prediktoren ab.
2. Nachträgliche Datengewinnung führt durch eingeschränkte Verfügbarkeit von geeigneten Prediktoren zu Feldfehlerprognosen schlechter Qualität.
3. Die Rahmenbedingungen für die Feldfehlerprognose müssen bereits zu Beginn der Software-Entwicklung festgelegt werden, um Feldfehlerprognosen hoher Qualität zu erhalten.

Sowohl durch die im Kapitel Grundlagen der Feldfehlerprognose von verschiedenen Autoren durchgeführten Studien als auch durch die im Rahmen der eigenen Fallstudie gewonnenen Erkenntnisse können alle drei Hypothesen eindeutig bestätigt werden.

Vor allem die teilweise guten, teilweise aber schlechten und darüber hinaus nicht einschätzbaren Ergebnisse der Fallstudie haben gezeigt, dass mit wenigen einfachen und nachträglich gewonnenen Metriken bzw. Prediktoren keine verwendbaren Ergebnisse erzielt werden können. Es hat sich auch gezeigt, dass eingeschränkte Trainingsdaten-Sets bzw. Trainingsdaten-Sets mit stark voneinander abweichenden Trainingsdatensätzen[17] die Ergebnisse der Modellierung verschlechtern.

Zur sinnvollen Verwendung der Feldfehlerprognose im Rahmen des realen Projektumfeldes, aus dem die Fallstudie entnommen ist, müssten alle Empfehlungen aus dem Kapitel Anwendungsempfehlungen umgesetzt werden. Aus Gründen der damit verbundenen Kosten wäre die Beschaffung von entsprechenden Software-Werkzeugen zur Gewinnung von Produktmetriken aus PowerBuilder-, Cobol-, MagnaX- und PL/SQL-Code wahrscheinlich aber nicht möglich.

---

[17] Vergleiche [Boet05]

Wäre eine breitere Palette von Metriken und Fehlerdaten auch aus der Vergangenheit verfügbar, könnte man erneut eine Modellfindung starten. Dabei wäre durchaus interessant, wie sich das Lineare Regressionsmodell mit Modellauswahl im Vergleich mit Algorithmen aus dem Bereich des maschinellen Lernens, wie z.B. Decision-Tree-Learner oder auch neuronale Netze, schlägt.

Literaturverzeichnis

[Bai03] Chenggang Bai, Kai-Yuan Cai, T.Y. Chen, An Efficient Defect Estimation Method for Software Defect Curves, COMPSAC'03, IEEE, 2003

[Biff00] Stefan Biffl, Thomas Grechenig, Monika Köhle, Evaluation of Inspectors' Defect Estimation Accuracy for a Requirements Document after Individual Inspection, IEEE, 2000

[Biff00a] Stefan Biffl, Using Inspection Data for Defect Estimation, IEEE Software November/December 2000, pp. 36 – 43, IEEE, 2000

[Biff01] Stefan Biffl, Wilfried Grossmann, Evaluating the Accuracy of Defect Estimation Models Based on Inspection Data From Two Inspection Cycles, IEEE, 2001

[Biff02] Stefan Biffl, Walter J. Gutjahr, Using a Reliability Growth Model to Control Software Inspection, Emperical Software Engineering, 7, pp. 257 – 284, Kluwer Academic Publishers, Boston, 2002

[Biff03] Stefan Biffl, Michael Halling, Sabine Köszegi, Investigating the Accuracy of Defect Estimation Models for Individuals and Teams Based on Inspection Data, ISES E'03, IEEE, 2003

[Boet05] Gary D. Boetticher, Nearest Neighbor Sampling for Better Defect Prediction, PROMISE'05, ACM, St. Louis, 2005

[Chal05] Venkata U. B. Challagulla, Farokh B. Bastani, I-Ling Yen, Empirical Assessment of Machine Learning based Software Defect Prediction Techniques, WORDS'05, IEEE, 2005

[Chan05] A Defect Estimation Approach for Sequential Inspection Using a Modified Capture-Recapture Model, COMPSAC'05, IEEE, 2005

[Hart00] Donald E. Harter, Mayuram S. Krishnan, Sandra A. Slaughter, Effects of Process Maturity on Quality, Cycle Time and Effort in Software Product Development, Management Science, Vol. 46, Nr. 4, 2000

[Jone99] Wendell Jones, John Hudepohl, Taghi Koshgoftaar, Edware Allen, Applications of a Usage Profile in Software Quality Models, 3rd European Conference on Software Maintenance and Reengineering, IEEE, Amsterdam, 1999

[Knab06] Patrick Knab, Martin Pinzger, Abraham Bernstein, Predicting Defect Densities in Source Code Files with Decision Tree Learners, MSR'06, ACM, Shanghai, 2006

[Koru05] A. Güneş Koru, Hongfang Liu, Building Effective Defect-Prediction Models in Practice, IEEE Software November/December 2005, pp. 23 – 29, IEEE, 2005

[Kosh92] Taghi Koshgoftaar, Bibhuti Bhattacharyya, Gary Richardson, Predicting Software Errors, During Development, Using Nonlinear Regression

Models: A Comparative Study, IEEE Tr. On Reliability, Vol. 41, Nr. 3, 1992

[Kosh92a] Taghi Koshgoftaar, Abhijit Pandya, Hamant More, A Neural Networks Approach for Predicting Software Development Faults, ISSRE, IEEE, 1992

[Kosh93] Taghi Koshgoftaar, John Munson, David Lanning, A Comparative Study of Predictive Models for Program Changes during System Testing and Maintenance, ICSM, IEEE, Quebec, 1993

[Kosh95] Taghi Koshgoftaar, Abhijit Pandya, David Lanning, Application of Neural Networks for Predicting Program Fault, Annals of Software Engineering, Vol. 1, 1995

[Khos01] Taghi Khoshgoftaar, Edward Allen, Jianyu Deng, Controlling Overfitting in Software Quality Models: Experiments with Regression Trees and Classification, METRICS, IEEE, London, 2001

[Kosh02] Taghi Koshgoftaar, Naeem Seliya, Tree-based Software Quality Estimation Models for Fault Prediction, METRICS, IEEE, Ottawa, 2002

[Li04] Paul Luo Li, Mary Shaw, James Herbsleb, Bonnie Ray, P. Santhanam, An Empirical Comparison of Defect Projection Models for Widely-deployed Production Software Systems, SIGSOFT, ACM, 2004

[Li04a] Paul Luo Li, Mary Shaw, Jim Herbsleb, Bonnie Ray, P. Santhanam, An Empirical Comparison of Field Defect Modeling Methods, FSE, Vol. 29, Nr. 6, 2004

[Li06] Paul Luo Li, James Herbsleb, Mary Shaw, Brian Robinson, Experiences and Results from Initiating Field Defect Prediction and Product Test Prioritization Efforts at ABB Inc., ICSE, ACM, Shanghai, 2006

[Li06a] Paul Luo Li, Mary Shaw, Jim Herbsleb, Bonnie Ray, P. Santhanam, An Empirical Comparison of Field Defect Modeling Methods, CMU Tech Report, CMU-ISRI-06-102, 2006

[Mock03] Audris Mockus, David Weiss, Ping Zhang, Understanding and Predicting Effort in Software Projects, ICSE, ACM, 2003

[Mock05] Audris Mockus, Ping Zhang, Paul Luo Li, Predictors of Customer Perceived Quality, ICSE, ACM, 2005

[Moha02] Saniay Mohapatra, B. Mohanty, Defect Prevention through Defect Prediction: A Case Study at Infosys, 2001 Proceedings IEEE International Conference on Software Maintenance, pp. 260 – 272, IEEE, Florence, 2001

[Naga05] Nachiappan Nagappan, Thomas Ball, Static Analysis Tools as Early Indicators of Pre-Release Defect Density, ICSE'05, ACM, St. Louis, 2005

[Padb02] Frank Padberg, Empirical Interval Estimates for the Defect Content After an Inspection, ICSE'02, ACM, Orlando, 2002

[Ragg02] Thomas Ragg, Frank Padberg, Ralf Schoknecht, Applying Machine Learning to Solve an Estimation Problem in Software Inspections, ICANN 2002, LNCS 2415, pp. 516 – 521, Springer, 2002

[Refo03] Marek Reformat, A Fuzzy-Based Meta-model for Reasoning about the Number of Software Defects, IFSA 2003, LNAI 2715, pp. 644 – 651, Springer-Verlag, 2003

[Rune98] Per Runeson, Claes Wohlin, An Experimental Evaluation of an Experience-Based Capture-Recapture Method in Software Code Inspections, Empirical Software Engineering, 3, pp. 381 – 406, Kluwer Academic Publishers, Boston, 1998

[Sany92] Shanker Sanyal, Kein Aida, Kostas Gaitanos, George Wowk, Sam Lahiri, Defect Tracking And Reliability Modeling For A New Product Release, Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative research - Volume 1, ACM, Toronto, 1992

[Sher05] Mark Sherriff, Nachiappan Nagappan, Laurie Williams, Mladen Vouk, Early Estimation of Defect Density Using an In-Process Haskell Metrics Model, A-MOST05, ACM, St. Louis, 2005

[Thel04] Thomas Thelin, Team-based Fault Content Estimation in the Software Inspection Process, ICSE'04, IEEE, 2004

[Wang06] Hao Wang, Fei Peng, Chao Zhang, Software Project Level Estimation Model Framework based on Bayesian Belief Networks, QSIC'06, IEEE, 2006

[Wolf06] Hans Peter Wolf, Peter Naeve, Veith Tiemann, Statistik aktiv mit R, UVK Verlagsgesellschaft mbH, Konstanz, 2006