Arbeitsgruppe Software Systems Engineering
Institut für Informatik
Ruprecht-Karls-Universität Heidelberg

Fehlerhäufigkeiten in objektorientierten Systemen: Basisauswertung einer Online-Umfrage

Technical Report SWEHD-TR-2006-01

Lars Borner, Universität Heidelberg, Arbeitsgruppe Software Systeme
Falk Fraikin, IT Transfer Office, Technische Universität Darmstadt
Matthias Hamburg, PSI Information Management GmbH, Düsseldorf
Stefan Jungmayr, Teradyne Diagnostic Solutions, Ismaning
Andreas Schönknecht, TUI InfoTec GmbH, Hannover

Version 1.0
Februar, 2006

Eine Publikation der Arbeitsgruppe Software Systems Engineering

Historie

Version	Datum	Grund der Änderung
V.1.0	20.02.2006	Ersterstellung

Die Arbeitsgruppe „Software Systems Engineering“ ist Teil des Instituts für Informatik der Ruprecht-Karls-Universität Heidelberg. Sie wird geleitet von

Prof. Dr. Barbara Paech
Institut für Informatik
Neuenheimer Feld 348
69120 Heidelberg

paech@informatik.uni-heidelberg.de
http://www-swe.informatik.uni-heidelberg.de/

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abstract

Für einen effizienten Einsatz von Qualitätssicherungsmaßnahmen ist die Kenntnis über die häufigsten Fehlerursachen entscheidend. Dieser Artikel beschreibt die Ergebnisse einer Online-Umfrage zu Fehlerhäufigkeiten in objektorientierten Systemen. Die Umfrage wurde vom Arbeitskreis „Testen objektorientierter Programme“ der GI (Gesellschaft für Informatik) mit 1219 Teilnehmern im Zeitraum vom 19. August bis 31. Oktober 2005 durchgeführt.

Kapitel 1
Motivation und Fokus

Die Kenntnis der häufigsten Fehlerursachen in einem gegebenen Testkontext ist für die Wahl von effizienten Qualitätssicherungsmaßnahmen entscheidend. Sind z.B. Fehler im Kontrollfluss eine häufige Fehlerkategorie, so sollte ein Testverfahren wie die Anweisungs- und Zweigüberdeckung angewendet werden, treten Datenflussfehler sehr häufig auf, so sollte eine Datenflussanalyse durchgeführt werden [Spil05].

Im Arbeitskreis „Testen objektorientierter Programme“ der Gesellschaft für Informatik [Url:TOOP] haben wir in der Literatur nach Listen bzw. Statistiken von Fehlerhäufigkeiten gesucht und solche u.a. [Beiz90], [Bind00], [Fent97], [Myer04] und [Rope94] gefunden.

Exkurs: „Die ganze Geschichte“
Der eigentliche Ausgangspunkt für die Umfrage war die Planung für eine empirische Untersuchung zum Einsatz von Testmethoden im XP-Umfeld. Für die empirische Untersuchung der Effektivität der Testmethoden sollte Software mit künstlichen Fehlern geimpft werden. Damit die Untersuchung aussagekräftig ist, sollten die künstlichen Fehler bzgl. Art und Verteilung dem Vorkommen in der betrieblichen Praxis entsprechen. Daraus folgte die Frage nach der Fehlerhäufigkeit in objektorientierten Systemen.

Die Literatur-Suche hat nur unzureichende Daten über Fehlerhäufigkeiten hervorgefördert. Die darin untersuchten Systeme und Fehlerkategorien deckten objektorientierte Software nicht ausreichend ab. Um diese Lücke zu schließen, wurde die Idee zu einer Online-Umfrage unter Testern und Software-Entwicklern geboren.

Der Fokus der Umfrage lag auf Fehler im Source-Code. Fehler in der Dokumentation oder im Entwicklungsprozess wurden bewusst in der Umfrage nicht berücksichtigt. Ebenso wurden Fehler bezüglich nicht-funktionaler Qualitätskriterien wie Benutzbarkeit, Wartbarkeit, Erweiterbarkeit und Zuverlässigkeit nicht betrachtet.

Die folgenden Abschnitte beschreiben die Umfrage und geben eine erste Auswertung der Ergebnisse wieder.

Kapitel 2
Fragebogen

Der Online-Fragebogen wurde in zwei Teile gegliedert. Im ersten Teil wurden die Teilnehmer- und Projektdaten, sowie die Häufigkeiten der Fehler-Hauptkategorien abgefragt. Dabei umfassten die Teilnehmerdaten die Testerfahrung und die Position der Teilnehmer. Die Informationen zu den verschiedenen Projekten beschränkten sich auf die eingesetzte Programmiersprache und die Branche, für die Software entwickelt und getestet wurde. Bei der Bewertung der Häufigkeiten der Fehlerkategorien standen den Teilnehmern im ersten Teil folgende sieben Fehler-Hauptkategorien zur Auswahl:

- Fehlerhafte Interpretation der Anforderungen
- Fehler im Kontrollfluss
- Fehler in der Berechnung
- Fehler in Klassen oder Datentypen
- Fehler im Datenfluss/Objektzugriff
- Schnittstellenfehler
- Konfigurationsfehler

Neben der Bewertung der Häufigkeiten der Fehler-Hauptkategorien hatten die Teilnehmer im ersten Teil noch die Möglichkeit eine eigene Fehler-Hauptkategorien und deren Häufigkeit anzugeben.

Im zweiten Teil wurden zu den sieben Fehler-Hauptkategorien jeweils 4 bis 10 Fehler-Unterkategorien vorgegeben, die von den Teilnehmern nach ihrer Häufigkeit bewertet werden konnten.

Die Definition der Fehlerunterkategorien orientierte sich an den Literatur-Angaben ([Beiz90], [Bind00], [Myer04]) und den persönlichen Erfahrungen der Arbeitskreis-Mitarbeiter.

Damit die Teilnehmer nicht die Häufigkeit für alle 7 Fehler-Hauptkategorien und 40 Fehler-Unterkategorien ausfüllen mussten, wurden ihnen zunächst die Hauptkategorien zur Bewertung angeboten (siehe Abbildung 23). Anschließend wurden nur die Unterkategorien zu den am höchsten bewerteten Hauptkategorien aufgerufen (siehe Abbildung 24).

Somit wurden die Fehler-Unterkategorien durchschnittlich von ca. der Hälfte der Teilnehmer bewertet.

Die Häufigkeit der Fehlerkategorien wurde anhand der folgenden Skala bewertet:

- nie
- selten
- oft
- sehr oft
- keine Angabe

Als Hilfe zu dieser Skala wurden Intervalle zur relativen Häufigkeit in Prozent angegeben. Die Skala und die zugeordneten Häufigkeitsintervalle sind in Tabelle 1 dargestellt.

Häufigkeit	Anteil der Fehlerkategorie an Gesamtfehleranzahl bzw. Anteil der Fehler-Unterkategorie an Fehler-Hauptkategorie
Nie	kleiner als 0,2%
Selten	größer gleich 0,2% (und kleiner als 7%)
Oft	größer gleich 7% (und kleiner 20%)
Sehr oft	größer gleich 20%

Tabelle 1 Definition der verwendeten Skala für Fehlerhäufigkeiten

Der Online-Fragebogen wurde von Lars Borner mit Servlets realisiert und am Lehrstuhl „Software-Systeme“ der Universität Heidelberg gehostet [Url:Umfr].

Kapitel 3
Durchführung der Umfrage

Die Umfrage wurde durch die Zeitschrift ObjektSPEKTRUM [Url:OS], die Computer-Zeitung [Url:CZ] sowie einen News-Ticker in heise.de [Url:heise] angekündigt. 1219 Teilnehmer hatten vom 19. August bis zum 31. Oktober 2005 die Möglichkeit genutzt, an der Umfrage teilzunehmen. (Die Teilnahmebedingungen sind in 8 wiedergegeben.)

Als Anreiz für die Teilnahme an der Umfrage wurden vier Test-Fachbücher als Gewinne durch den Verlag dpunkt zur Verfügung gestellt: [Link05] [Spil05] [Thal02] und [Vige04].

Die öffentliche Ziehung der Gewinner ist beim 23. Treffen der GI-Fachgruppe „Test, Analyse und Verifikation“ [Url:TAV] am 17. und 18. November 2005 in München sowie beim Seminar „Effiziente Testmethoden“ an der Universität Heidelberg am 13.12.2005 erfolgt[1]. Die Gewinner der Umfrage wurden auf der Homepage des Arbeitskreises [Url:TOOP] sowie im Tagungsbericht des 23. TAV-Treffens [Born06] bekannt gegeben.

[1] Eine zweite Ziehung wurde notwendig, weil zwei der vier in München gezogenen Gewinner keine gültige Email-Adresse als Kontakt angegeben hatten.

Kapitel 4

Auswertung der Teilnehmer- und Projekt-Daten

Als Teilnehmer- und Projektdaten wurden die Testerfahrung, die Position, die Branche, in der die Software erstellt wurde, sowie die eingesetzte Programmiersprache erfragt.

4.1 Branche

Die meisten Teilnehmer der Umfrage arbeiten für Software-Firmen, gefolgt von Dienstleistungsunternehmen und Unternehmen im technischen Bereich (siehe Tabelle 2 und Abbildung 1).

Branche	Absolut	Prozent
Software	537	44,05
Dienstleistung	348	28,55
technischer Bereich	198	16,24
kommerzieller Bereich	60	4,92
öffentlicher Bereich	42	3,45
k.A.	34	2,79
Gesamt	1219	100,00

Tabelle 2 Branche

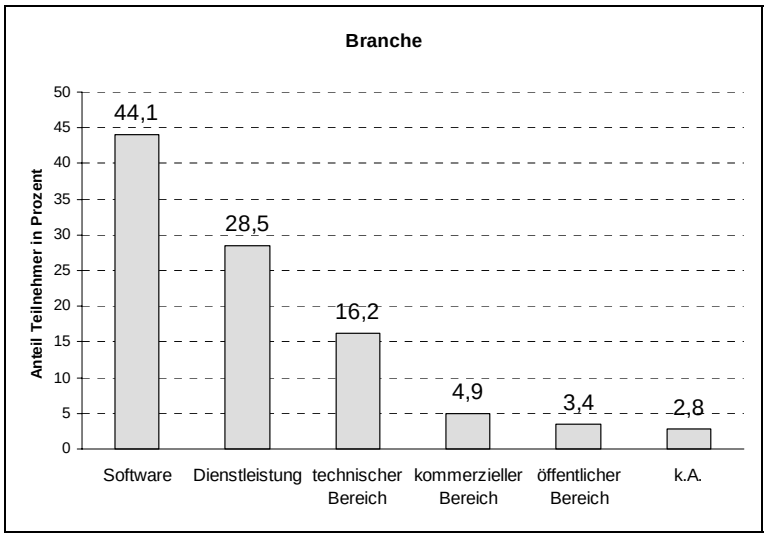


Abbildung 1 Branche – nach Häufigkeit sortiert

4.2 Testerfahrung

Die Mehrheit der Teilnehmer hatte eine Testerfahrung zwischen einem und fünf Jahren angegeben. (siehe Tabelle 3 und Abbildung 2).

Testerfahrung	Absolut	Prozent
Keine	79	6,48
1 -2 Jahre	359	29,45
3 - 5 Jahre	389	31,91
5 - 10 Jahre	239	19,61
mehr als 10 Jahre	121	9,93
k.A.	32	2,63
Gesamt	1219	100,00

Tabelle 3 Testerfahrung

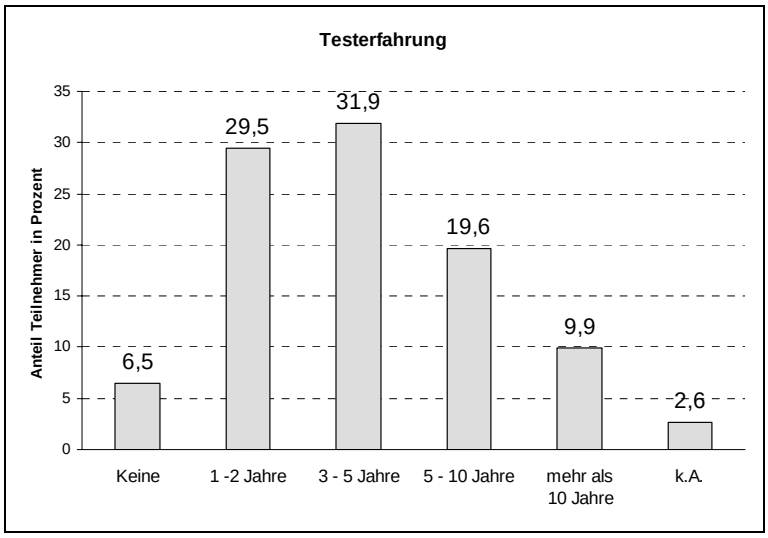


Abbildung 2 Testerfahrung – sortiert nach steigender Dauer der Erfahrung

4.3 Position

Die überwiegende Mehrheit der Teilnehmer ist als Entwickler bzw. Architekt tätig. Nur ein sehr kleiner Prozentsatz (2,54%) arbeitet als Tester (Tabelle 4 und Abbildung 3).

Position	Absolut	Prozent
Entwickler/Architekt	878	72,03
Projektleiter	143	11,73
sonstige Position	108	8,86
Tester	31	2,54
QM –Verantwortlicher	29	2,38
k.A.	30	2,46
Gesamt	1219	100

Tabelle 4 Position

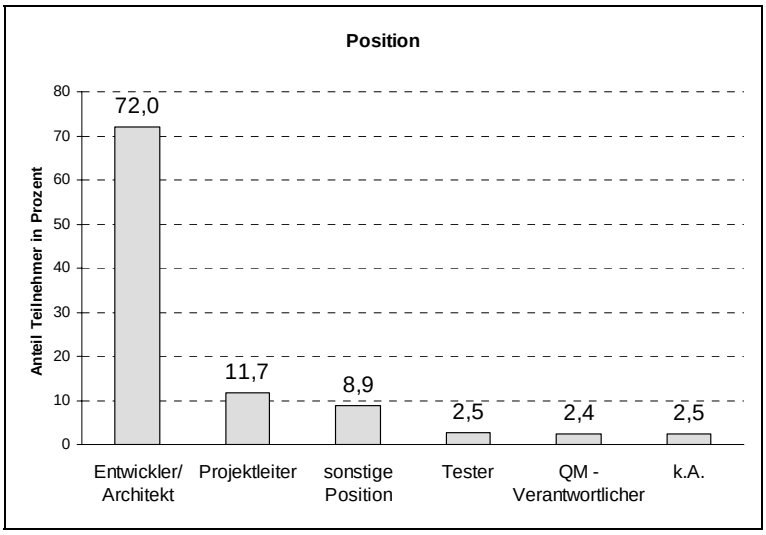


Abbildung 3 Position – sortiert nach Häufigkeit

4.4 Programmiersprache

Die in den Projekten der Teilnehmer am häufigsten eingesetzte Programmiersprache ist Java mit 40,94%, gefolgt von C++ mit 27,81% (Tabelle 5 und Abbildung 4).

Programmiersprache	Absolut	Prozent
Java	499	40,94
C++	339	27,81
andere	200	16,41
C#	91	7,47
Delphi	59	4,84
k.A.	31	2,54
Gesamt	1219	100

Tabelle 5 Programmiersprache

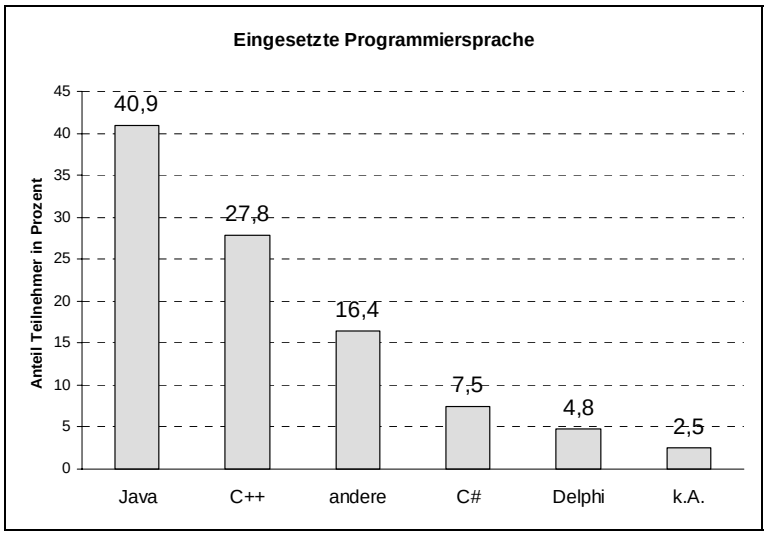


Abbildung 4 Eingesetzte Programmiersprache – sortiert nach Häufigkeit

Kapitel 5

Auswertung der Fehlerkategorien

Abbildung 5 zeigt die Fehler-Hauptkategorien und ihre Beschreibungen.

1	**Fehlerhafte Interpretation der Anforderungen** Kundenanforderung nicht implementiert, Implementierung nicht durch Kunde gefordert, Kundenanforderung falsch verstanden.
2	**Fehler im Kontrollfluss** Fehler in der Steuerung des dynamischen Ablaufs.
3	**Fehler in der Berechnung** Fehler in Berechnung und Manipulation von Variablen, Attributwerten und Ausdrücken.
4	**Fehler in Klassen oder Datentypen** Fehlerhafte Deklaration/Transformation des Typs bzw. der statischen Struktur von Klassen/Datentypen.
5	**Fehler im Datenfluss/Objektzugriff** Fehlerhafte Reihenfolge des Attribut-/Objektzugriffs, Fehler durch ungeeignete Objektzustände, dynamische objektübergreifende Abläufe.
6	**Schnittstellenfehler** Fehler durch falsche Verwendung einer Klassen-/Komponentenschnittstelle.
7	**Konfigurationsfehler** Falsche Konfiguration des Builds oder des laufenden Systems.

Abbildung 5 Fehler-Hauptkategorien

Die am häufigsten genannte Fehler-Hauptkategorie ist „fehlerhafte Interpretation der Anforderungen“. Fast 60% der Teilnehmer haben sie mit „sehr oft“ oder „oft“ gewertet. Die Hauptkategorie „Fehler in Klassen oder Datentypen“ wurde hingegen selten, von knapp über 20% aller Teilnehmer mit „sehr oft“ oder „oft“ gewertet. Alle anderen Hauptkategorien liegen im Mittelfeld, mit der Nennung „sehr oft“ oder „oft“ um 40% (Tabelle 6 und Abbildung 6).

	1 Fehlerhafte Interpretation der Anforderungen	2 Fehler im Kontrollfluss	3 Fehler in der Berechnung	4 Fehler in Klassen oder Datentypen	5 Fehler im Datenfluss/ Objektzugriff	6 Schnittstellen-fehler	7 Konfigurations-fehler
nie	43	31	74	218	110	167	167
selten	439	646	743	728	558	654	557
oft	479	440	332	218	407	287	348
sehr oft	232	68	56	30	114	90	125
k.A.	26	34	14	25	30	21	22
Gesamt	1219	1219	1219	1219	1219	1219	1219

Tabelle 6 Fehler-Hauptkategorien

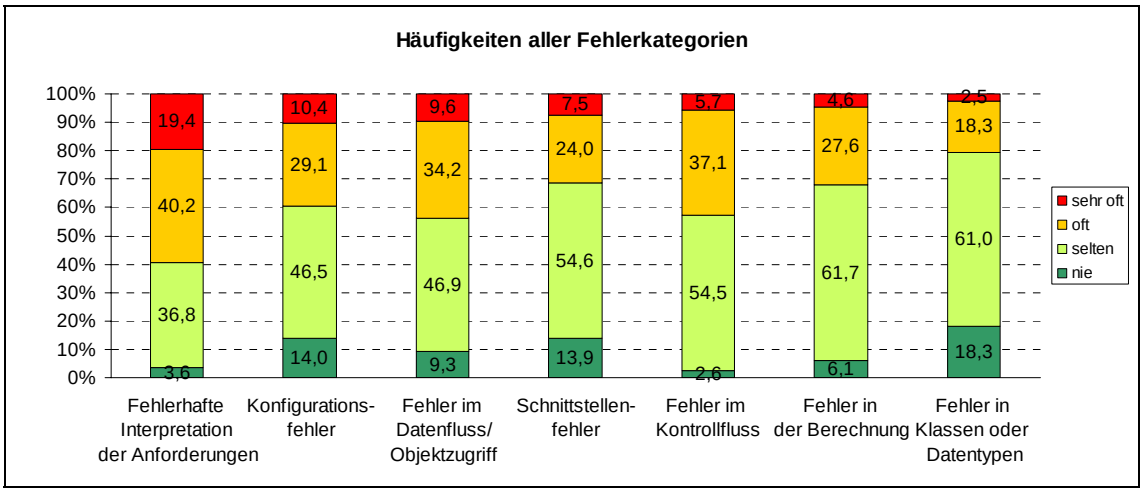


Abbildung 6 Fehler-Hauptkategorien – sortiert nach Anzahl der Antwort „sehr oft“.

Kapitel 6

Auswertung - Fehler-Unterkategorien

In diesem Kapitel stellen wir zu jeder Fehler-Hauptkategorie die Ergebnisse für die Fehler-Unterkategorien dar:

- die Definitionen der Fehler-Unterkategorien,
- eine Tabelle mit den absoluten Häufigkeiten, und
- ein Säulendiagramm mit den relativen Häufigkeiten in Prozent, in fallender Reihenfolge der Häufigkeit sehr oft".

6.1 Fehlerhafte Interpretation der Anforderungen

Abbildung 7 zeigt die Fehler-Unterkategorien und ihre Beschreibungen für die Fehler-Hauptkategorien „Fehlerhafte Interpretation der Anforderungen".

1	**Fehlerhafte Interpretation der Anforderungen** Kundenanforderung nicht implementiert, Implementierung nicht durch Kunde gefordert, Kundenanforderung falsch verstanden.
1.1	**Fehlende Implementierung einer expliziten Anforderung** Eine Funktionalität wurde nicht oder nur unvollständig realisiert. Beispiel: Anwendungsfall/Szenario vergessen oder unvollständig umgesetzt.
1.2	**Fehlende Implementierung einer impliziten Anforderung** Beispiele: Externer Ausnahmefall wie z.B. Netzwerkproblem, Druckerproblem etc. wird nicht wie erwartet behandelt.
1.3	**Implementierung ohne Vorliegen einer Anforderung** Eine Funktionalität oder Benutzerausgabe wurde realisiert, ohne dass eine Anforderung vorliegt.
1.4	**Kundenanforderung falsch verstanden** Kundenanforderung wurde falsch verstanden und daher nicht richtig implementiert.
1.5	**Fehler in Ausgabepräsentation** Berechnung richtig, aber Systemausgabe fehlerhaft. Beispiele: Bildschirmausgabe, Druckausgabe oder Fehlermeldung ist inkorrekt bzgl. Inhalt, Formatierung oder Rechtschreibung.

Abbildung 7 Fehlerkategorien - Fehlerhafte Interpretation der Anforderungen

Die fehlerhafte Interpretation von Anforderungen ist die am häufigsten genannte Fehler-Hauptkategorie. Die „fehlende Implementierung einer impliziten Anforderung“ ist dabei die am stärksten gewichtete Fehler-Unterkategorie der gesamten Erhebung (18,8% „sehr oft“ und 45,3 „oft“). Die „fehlende Implementierung einer expliziten Anforderung“ verfehlte als einzige Fehler-Unterkategorie knapp die 40%-Marke für „sehr oft“ und „oft“ (Tabelle 7 und Abbildung 8).

	1.1. Fehlende Implementierung einer expliziten Anforderung	1.2. Fehlende Implementierung einer impliziten Anforderung	1.3. Implementierung ohne Vorliegen einer Anforderung	1.4. Kundenanforderung falsch verstanden	1.5. Fehler in Ausgabe-präsentation
nie	116	45	78	32	88
selten	454	288	378	322	464
oft	304	421	373	425	295
sehr oft	59	175	101	152	79
k.A.	286	290	289	288	293
Gesamt	1219	1219	1219	1219	1219

Tabelle 7 Fehlerhafte Interpretation der Anforderungen

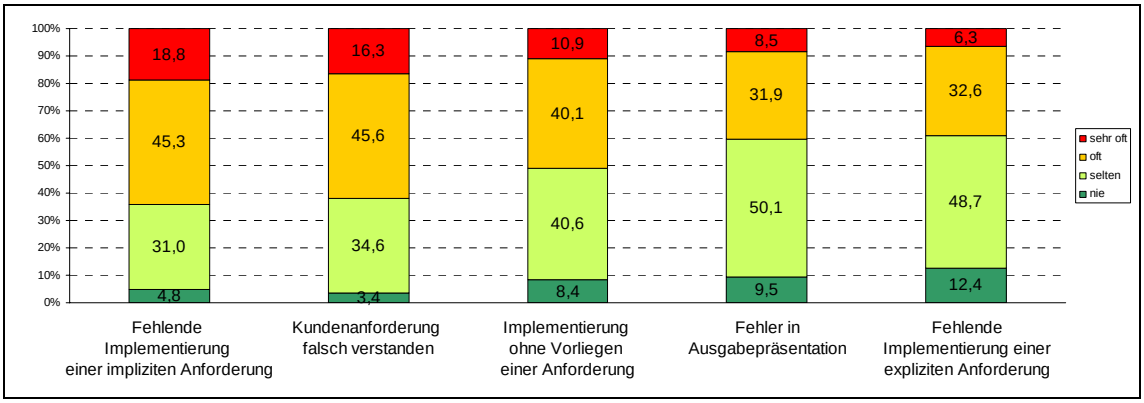


Abbildung 8 Fehlerhafte Interpretation der Anforderungen - sortiert nach „sehr oft“.

6.2 Fehler im Kontrollfluss

Abbildung 9 zeigt die Fehler-Unterkategorien und Beschreibungen für die Fehler-Hauptkategorie „Kontrollflussfehler“.

2	**Fehler im Kontrollfluss** Fehler in der Steuerung des dynamischen Ablaufs.
2.1	**Fehler in Prädikat** Fehlerhafte Logik in Verzweigungs- oder Schleifenend-Bedingung, die zu falschem Kontrollfluss führt. Beispiele: falscher boolescher Operator, falsche Klammerung von Operatoren.
2.2	**Fehler in Selektions-Konstrukt** Fehler in IF- oder Select-Konstrukt. Beispiele: fehlender Default-Fall, break vergessen, fehlende Blockbildung (geschweifte Klammern vergessen).
2.3	**Fehler in Schleifenkonstrukt** Beispiele: Fehlerhafter Initial- / Endwert bei FOR-Schleifen, falsche Inkrementierung, fehlerhafte Blockbildung.
2.4	**Fehler in Zustandslogik** Fehlerhafte oder fehlende Initialisierung des Objektzustands, fehlerhafter Zustandsübergang.
2.5	**Fehler in Behandlung von internem Ausnahmefall** Fehlerhafte Behandlung einer Exception, die von aufgerufener Komponente des eigenen Systems gemeldet wird. Beispiel: nicht behandelte Exception, finally vergessen.
2.6	**Methode auf falschem Objekt aufgerufen** Der Namen der aufgerufenen Methode ist richtig, aber der Aufruf geht nicht an das beabsichtigte Objekt. Beispiel: Klasse mit gleichem Namen aus falschem Paket aufgerufen.
2.7	**Falscher Methodenaufruf durch Polymorphie bzw. Überladen** Der Name der aufgerufenen Methode ist richtig, aber der Aufruf geht an eine nicht beabsichtigte Implementierung der Methode. Dies kann durch dynamische Bindung oder durch falsche Parametertypen bzw. -Anzahl verursacht werden.
2.8	**Falsche Methode ausgewählt** Mögliche Ursachen: irreführende Namensgebung der Methode; Name und Verantwortlichkeit der Methode nicht konsistent.
2.9	**Fehler bei parallelen Prozessen.** Fehler in der Verarbeitung von parallelen Prozessen. Beispiele: Race conditions
2.10	**Fehler in deklarativer Kontrollflusssteuerung** Beispiel: Dialogablaufsteuerung per XML.

Abbildung 9 Fehlerkategorien - Fehler im Kontrollfluss

Die Fehler-Unterkategorien „Fehler in Behandlung von internem Ausnahmefall“ (13,0% „sehr oft“ und 36,5 „oft“) und „Fehler bei parallelen Prozessen“ (10,6% „sehr oft“ und 31,8% „oft“) wurden sehr häufig genannt (Tabelle 8, Abbildung 10 und Abbildung 11).

	2.1. Fehler in Prädikat	2.2. Fehler in Selektions-Konstrukt	2.3. Fehler in Schleifenkonstrukt	2.4. Fehler in Zustandslogik	2.5. Fehler in Behandlung von internem Ausnahmefall	2.6. Methode auf falschem Objekt aufgerufen	2.7. Falscher Methodenaufruf durch Polymorphie bzw. Überladen	2.8. Falsche Methode ausgewählt	2.9. Fehler bei parallelen Prozessen.	2.10. Fehler in deklarativer Kontrollflusssteuerung
nie	72	177	158	92	103	458	332	242	109	214
selten	505	496	524	423	317	338	408	428	326	340
oft	224	152	138	278	303	40	79	160	240	66
sehr oft	41	22	24	48	108	6	8	18	80	14
k.A.	377	372	375	378	388	377	392	371	464	585
Gesamt	1219	1219	1219	1219	1219	1219	1219	1219	1219	1219

Tabelle 8 Fehler im Kontrollfluss

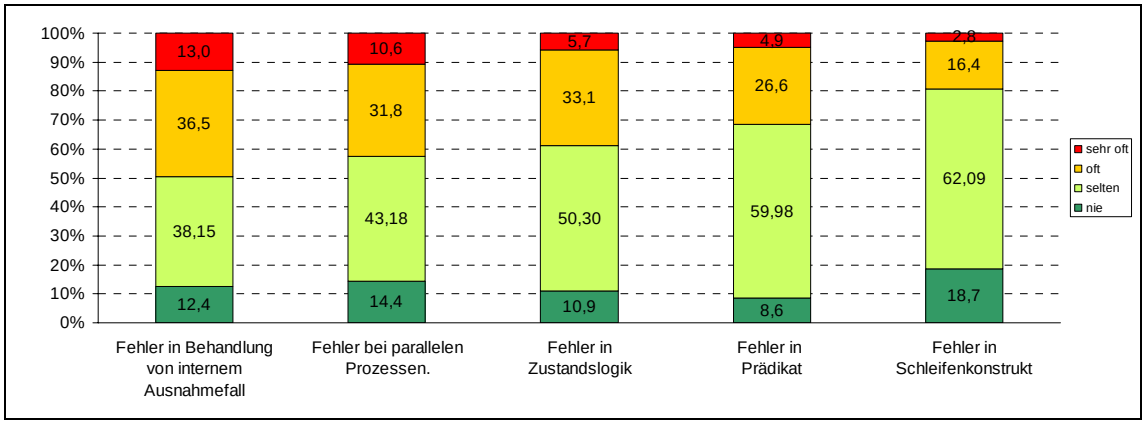


Abbildung 10 Fehler im Kontrollfluss– sortiert nach „sehr oft“, Teil A

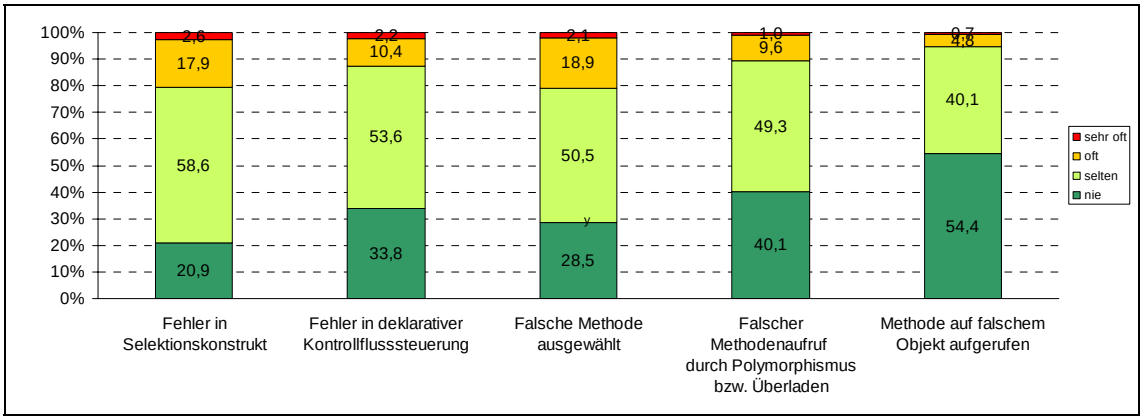


Abbildung 11 Fehler im Kontrollfluss – sortiert nach „sehr oft“, Teil B

6.3 Fehler in der Berechnung

Abbildung 12 zeigt die Fehler-Unterkategorien und Beschreibungen der Fehler-Hauptkategorie „Berechungsfehlern“.

3 Fehler in der Berechnung

Fehler in Berechnung und Manipulation von Variablen, Attributwerten und Ausdrücken.

3.1 Falscher Algorithmus ausgewählt

Fehlerhafter oder nicht adäquater Algorithmus ausgewählt (aber korrekt implementiert). Beispiel: Nutzung des falschen Näherungsalgorithmus; Nutzung eines Algorithmus, dessen Voraussetzungen nicht erfüllt sind.

3.2 Fehler in Arithmetik

Fehler in der Auswertung eines arithmetischen Ausdrucks (ausgenommen Zeigerarithmetik). Beispiel: Falscher Operator, falscher boolescher Ausdruck, fehlerhafte Berücksichtigung der Genauigkeit einer Berechnung.

3.3 Fehler in Zeigerarithmetik

Fehler bei der expliziten Berechnung einer relativen Speicheradresse.

3.4 Fehler in String-Manipulation

Fehler in Verarbeitung eines Text-Attributs. Beispiele: Fehler bzgl. Textoperationen wie Verknüpfung, Einfügung, Konvertierung in Großschreibung, String-Vergleich.

3.5 Fehler in Initialisierung

Fehlerhafte Initialisierung von Variable oder Attribut. Beispiele: Falscher Konstantenwert, falscher Konstruktoraufruf.

3.6	**Fehler in Ressourcen-Verwendung** Fehlerhafte Freigabe von temporären Daten, Systemressourcen, etc., Überlauffehler, fehlerhaftes Speicher-Management, fehlender Destruktor-Aufruf.

Abbildung 12 Fehlerkategorien – Fehler in der Berechnung

Als Fehler-Unterkategorien wurden „Fehler in Ressourcen-Verwendung“ (9,6% „sehr oft“ und 26,6 „oft“) und „Fehler in Zeigerarithmetik“ (4,3% „sehr oft“ und 24,5% „oft“) am häufigsten genannt (Tabelle 9 und Abbildung 13), sie blieben aber damit hinter den Fehler-Unterkategorien des Spitzenreiters „Fehlerhafte Interpretation der Anforderungen“ allesamt zurück.

	3.1 Falscher Algorithmus ausgewählt	3.2 Fehler in Arithmetik	3.3 Fehler in Zeigerarithmetik	3.4 Fehler in String-Manipulation	3.5 Fehler in Initialisierung	3.6 Fehler in Ressourcen-Verwendung
nie	249	135	337	139	153	154
selten	405	425	188	392	425	304
oft	76	159	91	183	150	191
sehr oft	6	27	47	32	23	69
k.A.	483	473	556	473	468	501
Gesamt	1219	1219	1219	1219	1219	1219

Tabelle 9 Fehler in der Berechnung

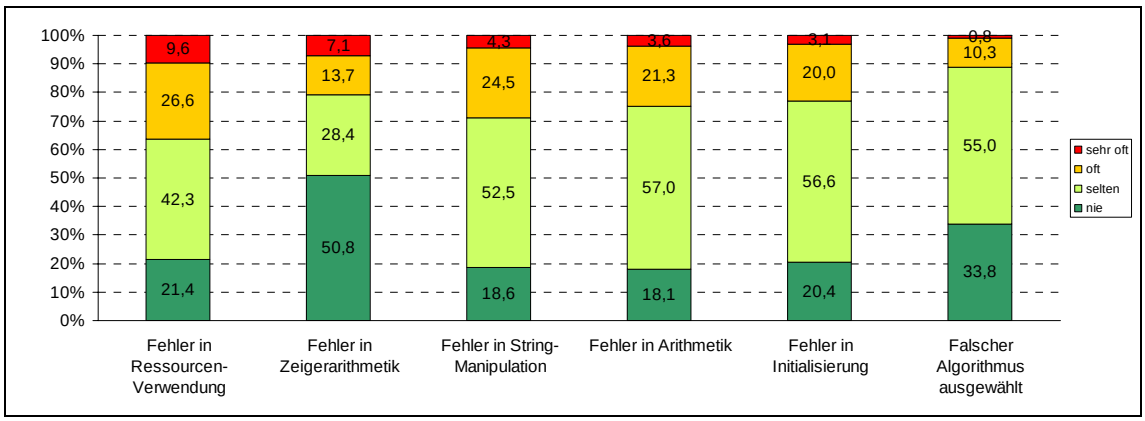


Abbildung 13 Fehler in der Berechnung – sortiert nach „sehr oft“

Fehler in der Zeigerarithmetik treten schwerpunktmäßig bei der Programmiersprache C++ auf (Tabelle 10 und Abbildung 14).

	Programmiersprache				
	C++	Java	C#	Delphi	Andere
nie	37	190	27	16	64
selten	97	42	10	9	25
oft	47	25	6	7	5
sehr oft	30	5	0	4	3
k.A.	128	237	48	23	103
gesamt	339	499	91	59	200

Tabelle 10 Programmiersprache und Zeigerarithmetik

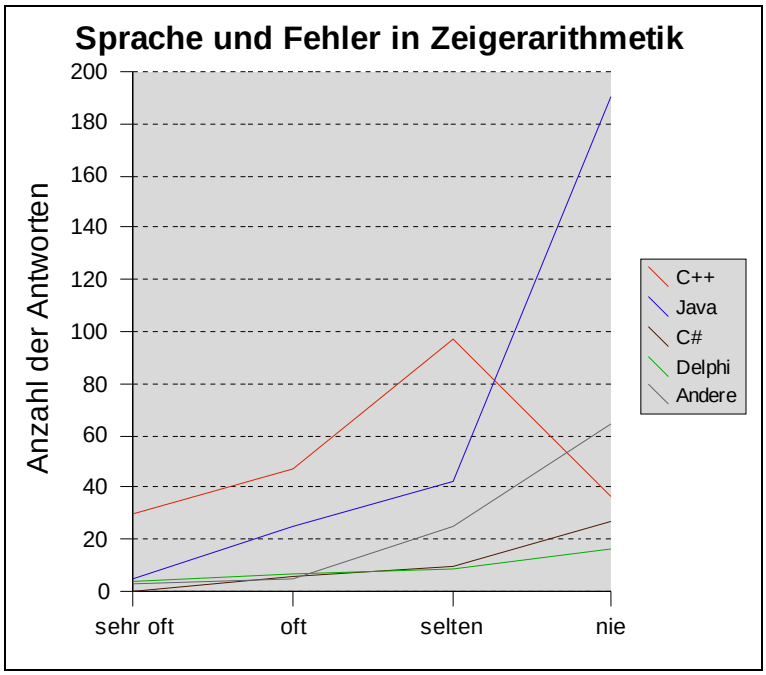


Abbildung 14 Programmiersprache und Zeigerarithmetik

6.4 Fehler in Klassen oder Datentypen

Abbildung 15 zeigt die Fehler-Unterkategorien und Beschreibungen der Fehler-Hauptkategorie „Fehlern in Klassen und Datentypen“.

4	**Fehler in Klassen oder Datentypen** Fehlerhafte Deklaration/Transformation des Typs bzw. der statischen Struktur von Klassen/Datentypen.
4.1	**Fehlerhafte Deklaration eines Attributs bzw. einer Variable** Der verwendete Datentyp ist nicht adäquat. Beispiele: Integer statt Float, Pointer statt Integer, Array statt Skalar, falsche Dimension.
4.2	**Fehler in Typtransformation** Objekt wird inkorrekter Typtransformation unterzogen. Notwendige Typtransformation vergessen. Beispiel: Float zu Integer, Pointer zu Integer, implizite Typtransformation übersehen, fehlgeschlagener Cast.
4.3	**Fehler durch Mehrfachvererbung** Probleme mit der Namensauflösung, Fehler durch Redefinition eines Attributs / einer Methode, Fehler durch (zu) hohe Komplexität der Klasse.
4.4	**Methode nicht implementiert/überschrieben** Beispiel: Fehlerhafte Einbindung in Rahmenwerk, das Implementieren oder Überschreiben einer Methode erfordert.
4.5	**Fehlerhafte Deklaration einer Methode / eines Attributes als statisch bzw. dynamisch.** Beispiel: Ein Attribut einer Klasse wird fälschlicherweise als statisch deklariert, so dass alle Objekte der Klasse auf der selben Variable arbeiten und nicht über ein jeweils eigenes Exemplar verfügen.

Abbildung 15 Fehlerkategorien – Fehler in Klassen oder Datentypen

Fehler in Typtransformationen wurden am häufigsten genannt (5,8% „sehr oft“ und 21,3 „oft“), die anderen Fehler-Unterkategorien wurden selten mit „sehr oft“ oder „oft“ bewertet (Tabelle 11 und Abbildung 16).

	4.1 Fehlerhafte Deklaration eines Attributs bzw. einer Variable	4.2 Fehler in Typtrans-formation	4.3 Fehler durch Mehrfach-vererbung	4.4 Methode nicht implementiert/ überschrieben	4.5 Fehlerhafte Deklaration einer Methode / eines Attributes als statisch bzw. dynamisch.
nie	158	114	207	168	240
selten	312	303	223	301	242
oft	93	122	92	74	61
sehr oft	15	33	16	17	16
k.A.	641	647	681	639	660
Gesamt	1219	1219	1219	1219	1219

Tabelle 11 Fehler in Klassen oder Datentypen

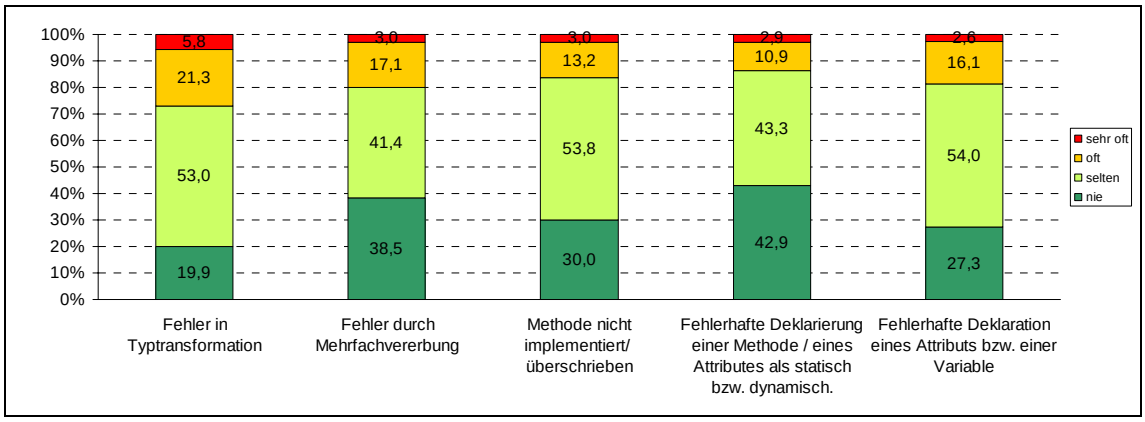


Abbildung 16 Fehler in Klassen oder Datentypen– sortiert nach „sehr oft“

6.5 Fehler im Datenfluss/Objektzugriff

Abbildung 17 zeigt die Fehler-Unterkategorien und ihre Beschreibungen für die Fehler-Hauptkategorie „Fehler im Datenfluss/Objektzugriff“.

5	**Fehler im Datenfluss/Objektzugriff** Fehlerhafte Reihenfolge des Attribut-/Objektzugriffs, Fehler durch ungeeignete Objektzustände, dynamische objektübergreifende Abläufe.
5.1	**Datenflussanomalie** Fehlerhafte Sequenz von Attribut- bzw. Objektzugriffen. Beispiel: Zugriff auf eine Variable oder ein Attribut bevor es erzeugt wurde (Null-Pointer Exception). Zweimalige Wertzuweisung ohne Verwendung des ersten Werts.
5.2	**Dead Lock** Sperr- oder Synchronisationsmechanismus nicht korrekt verwendet, so dass sich verschiedene Programmteile beim Ressourcenzugriff gegenseitig blockieren.
5.3	**Falsche Zugriffsart** Der Zugriff auf ein Objekt erfolgt in der falschen Art. Beispiele: Direkter Zugriff auf ein Objekt, für das indirekter Zugriff erforderlich ist; call-by-value anstatt call-by-reference.
5.4	**Fehler durch Umgehung des Zugriffsschutzes** Fehler im Einsatzfeld von Reflection (Introspektion), friend-Deklaration etc. Beispiele: Zugriff auf Methode, die privat ist.

Abbildung 17 Fehlerkategorien – Fehler im Datenfluss/Objektzugriff

Am häufigsten wurde die Fehler-Unterkategorie „Datenflussanomalie“ genannt: 13,1% „sehr oft“ und 36,7% „oft“ (Abbildung 22 und Tabelle 8).

	5.1 Datenfluss-anomalie	5.2 Dead Lock	5.3 Falsche Zugriffsart	5.4 Fehler durch Umgehung des Zugriffsschutzes
nie	79	191	251	396
selten	339	438	431	318
oft	306	142	121	75
sehr oft	109	34	16	15
k.A.	386	414	400	415
Gesamt	1219	1219	1219	1219

Tabelle 12 Fehler im Datenfluss/Objektzugriff

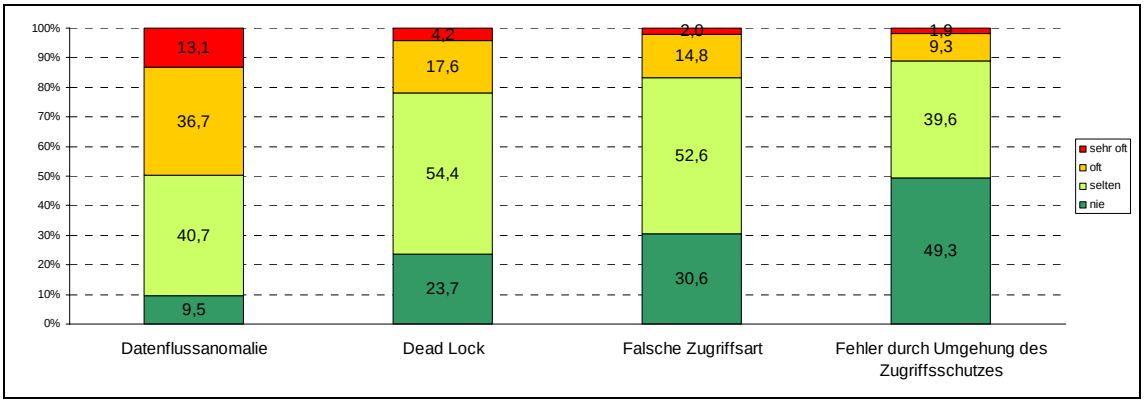


Abbildung 18 Fehler im Datenfluss/Objektzugriff – sortiert nach „sehr oft“

6.6 Schnittstellenfehler

Abbildung 19 zeigt die Fehler-Unterkategorien und Beschreibungen für die Fehler-Hauptkategorie „Schnittstellenfehler“.

6	**Schnittstellenfehler** Fehler durch falsche Verwendung einer Klassen-/Komponentenschnittstelle.
6.1	**Aufgerufene Klasse implementiert Schnittstelle nicht bzw. fehlerhaft** Beispiel: Aufruf einer nicht-vorhandenen Methode. Fehler ist insbesondere dann relevant, wenn er nicht durch den Compiler entdeckt wird, z.B. bei Einsatz von Middleware oder Reflektion.
6.2	**Fehler in übergebenen Parametern** Beispiele: Falscher Parametertyp, falscher Parameterwert, falsche Parameterreihenfolge, falsche Parameteranzahl.
6.3	**Fehler in der Interpretation von Rückgabewerten** Beispiele: Zugriff auf den falschen Rückgabewert, falscher Typ des Rückgabeparameters (z.B: XML-Datei) benutzt, Struktur des Rückgabeparameters falsch interpretiert.
6.4	**Aufruf bzw. Aufrufreihenfolge der Methode falsch** Aufruf im falschen Zustand, duplizierte Aufrufe, überflüssige Aufrufe, falsche Aufrufreihenfolge.
6.5	**Nichtbeachtung von Kontrakten** Vor-/Nachbedingung oder Invariante ist verletzt. Seiteneffekt wurde nicht berücksichtigt. Beispiele: Nicht-semantikerhaltende Vererbung; Instanzvariablen überschrieben.

Abbildung 19 Fehlerkategorien – Schnittstellenfehler

Unter den fünf Fehler-Unterkategorien gab es keine Ausreißer nach oben oder nach unten. Sie wurden jeweils von knapp 40% bzw. gut 20% der Teilnehmer mit „sehr oft“ und „oft“ bewertet (Tabelle 13 und Abbildung 20).

	6.1 Aufgerufene Klasse implementiert Schnittstelle nicht bzw. fehlerhaft	6.2 Fehler in übergebenen Parametern	6.3 Fehler in der Interpretation von Rückgabewerten	6.4 Aufruf bzw. Aufrufreihenfolge der Methode falsch	6.5 Nichtbeachtung von Kontrakten
nie	194	151	144	103	108
selten	358	346	365	367	306
oft	118	184	179	196	208
sehr oft	22	37	27	41	47
k.A.	527	501	504	512	550
Gesamt	1219	1219	1219	1219	1219

Tabelle 13 Schnittstellenfehler

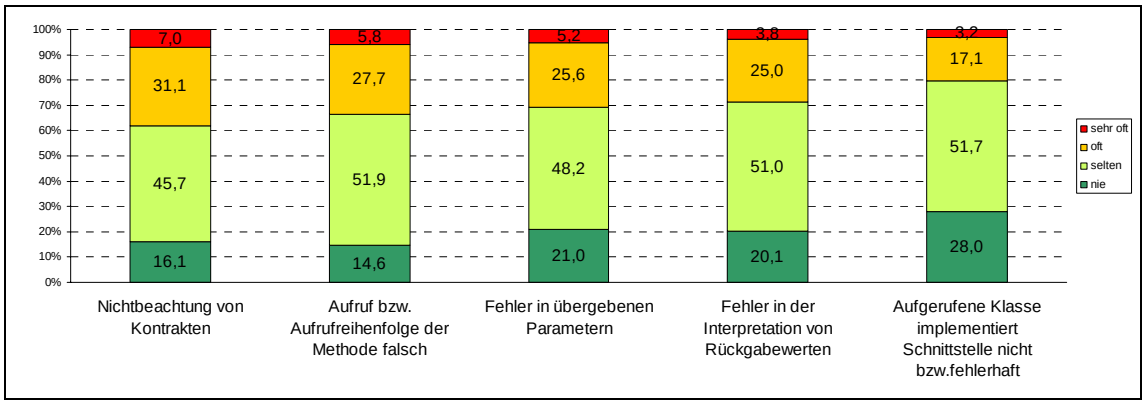


Abbildung 20 Schnittstellenfehler – sortiert nach „sehr oft“

6.7 Konfigurationsfehler

Abbildung 21 zeigt die Fehler-Unterkategorien und ihre Beschreibungen für Fehler-Hauptkategorie „Konfigurationsfehler“.

7	**Konfigurationsfehler** Falsche Konfiguration des Builds oder des laufenden Systems.
7.1	**Aufgerufene Komponente fehlt oder liegt in falscher Version vor** Die aufgerufene Komponente existiert nicht. Beispiele: falsche Source (falsche Version getestet), mit alter Version des Bytecode gearbeitet, inkompatible Versionen von Programmen, Laufzeitumgebungen.
7.2	**Fehler in Deployment-Deskriptoren** Fehler in deklarativer Konfigurationsdatei eines komponentenbasierten Systems.
7.3	**Umgebungsvariablen falsch gesetzt oder ignoriert** Das Laufzeitsystem kann notwendige Ressourcen nicht finden, da Pfadangaben nicht korrekt gesetzt bzw. verwendet werden.
7.4	**Testcode unbeabsichtigt im operativen System enthalten** Spezieller Testcode (z.B. zum Auslesen von Objektzuständen) ist im operativen System enthalten und stellt unerwünschte Funktionalität oder ein Performanzproblem dar. Ursachen: z.B. Fehler im Build-Prozess; Testcode wird von operativem Code verwendet und lässt sich nicht mehr einfach entfernen.
7.5	**Eigenschaften der Laufzeitumgebung(en) nicht beachtet.** Beispiele: Zeilenumbruch mit falscher Escape-Sequenz, unterschiedliche Datumsformate, falscher File-Separator.

Abbildung 21 Fehlerkategorien - Konfigurationsfehler

Am häufigsten wurden die Fehler-Unterkategorien „Aufgerufene Komponente fehlt oder liegt in falscher Version vor“ (9,0% „sehr oft“ und 32,0% „oft“) und „Umgebungsvariablen falsch gesetzt oder ignoriert“ (7,4% „sehr oft“ und 32,6% „oft“) genannt (Tabelle 14 und Abbildung 22).

	7.1 Aufgerufene Komponente fehlt oder liegt in falscher Version vor	7.2 Fehler in Deployment-Deskriptoren	7.3 Umgebungs-variablen falsch gesetzt oder ignoriert	7.4 Testcode unbeabsichtigt im operativen System enthalten	7.5 Eigenschaften der Laufzeitumgebung(en) nicht beachtet.
nie	109	150	124	217	156
selten	334	313	325	323	359
oft	241	156	244	158	190
sehr oft	68	46	55	51	46
k.A.	467	554	471	470	468
Gesamt	1219	1219	1219	1219	1219

Tabelle 14 Konfigurationsfehler

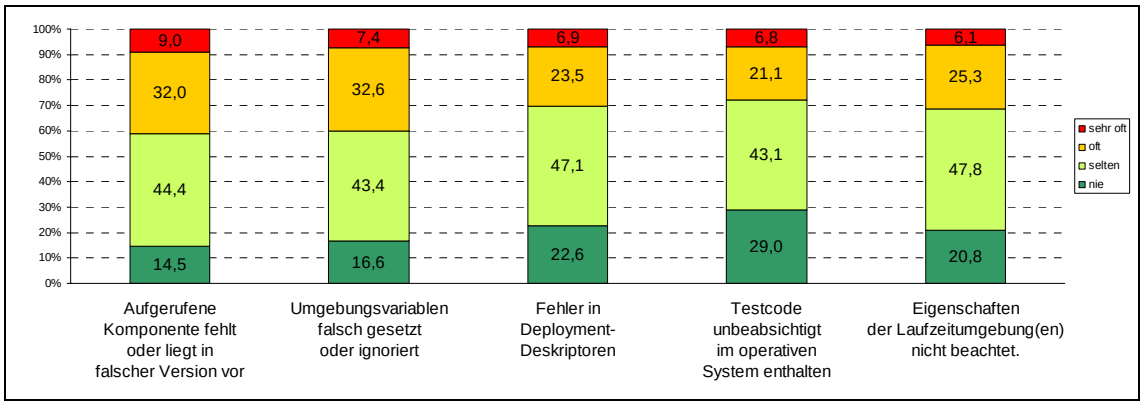


Abbildung 22 Konfigurationsfehler – sortiert nach „sehr oft“

6.8 Zusätzlich genannte Fehlerkategorien

Jeder Teilnehmer konnte eine zusätzliche Fehlerkategorie benennen und seine Häufigkeit bewerten. Von den 1219 Teilnehmern haben 215 von dieser Möglichkeit Gebrauch gemacht. Diese Nennungen wurden bei der statistischen Auswertung nicht berücksichtigt und im Folgenden diskutiert.

Die größte Gruppe (54%) der Nennungen betraf Fehlerkategorien, die bewusst nicht im Fokus der Umfrage lagen, wie z.B.

- Fehler in der Hilfe,
- Fehler in den Anforderungs-, Entwurfs- und Architekturdokumenten sowie in der Programmdokumentation,
- Fehler bezüglich Software-Qualitätsmerkmalen wie Änderbarkeit, Wartbarkeit, Benutzbarkeit, Performanz,
- Prozessprobleme, wie z.B. Zeitdruck, Qualifikation, oder Qualitätssicherung von Zulieferern,
- Testfehler,
- Fehler in Fremdsoftware, z.B. in verwendeten Klassenbibliotheken oder Frameworks, in der Entwicklungsumgebung oder im Compiler.

Ohne die Bedeutung solcher Fehler für die Software-Entwicklung in Frage zu stellen, haben wir diese Kategorien in unserer Klassifikation ausgeklammert, da sie nicht direkt Fehler im Source-Code von objektorientierten Programmen darstellen.

Eine zweite große Gruppe (41%) der von den Teilnehmern zusätzlich genannten Fehlerkategorien hätte den existierenden Kategorien zugeordnet werden können:

- Memory Leaks bzw. fehlende Speicherfreigabe (gehören zu Kategorie 3.6),
- Unkorrekte Fehlerbehandlung bzw. Exception-Handling (gehört zu Kategorie 2.5),
- Fehler durch Nebenläufigkeit bzw. Parallelität (gehören zu Kategorie 2.9).

Diese Nennungen liefern Hinweise auf mögliche Verbesserungen der Beschreibung der Fehlerkategorien in zukünftigen Umfragen.

Ein kleinerer Teil der Nennungen (5%) war zu unspezifisch bzw. konnte nicht richtig interpretiert werden wie „programmiersprachenspezifische Fehler“ oder „Logikfehler bei der Verwendung von Objektspinnen“.

Die Auswertung der von den Teilnehmern zusätzlich genannten Fehlerkategorien hat insgesamt keine nennenswerten neuen Fehlerkategorien im Vergleich zu den von den Autoren vorgegebenen Fehlerkategorien aufgedeckt. Dies deutet darauf hin, dass die die potentiell in der Praxis auftretenden Fehler weitestgehend abdecken, wenn auch die Erläuterungen in wenigen Einzelfällen verbessert werden können.

Kapitel 7
Zusammenfassung

An der Umfrage haben sich erfreulich viele Teilnehmer beteiligt (insbesondere dank der Verlautbarung in [Url:heise]). Daraus lässt sich vermutlich auch die Tatsache erklären, dass die Teilnehmer hauptsächlich Entwickler waren und nicht Tester.

Die folgende Liste beschreibt die wichtigsten Resultate der Umfrage. Dabei werden die Fehler-Hauptkategorien in absteigender Anzahl der Häufigkeitsangabe „sehr oft" genannt:

1. *Fehlerhafte Interpretation von Anforderungen*: Die fehlende Implementierung von impliziten Anforderungen sowie falsch verstandene Kundenanforderungen sind die Fehlerkategorien mit der größten Häufigkeit.
2. *Konfigurationsfehler*: Das Fehlen der aufgerufenen Komponente bzw. das Vorliegen einer falschen Version wurde hier als häufigste Fehlerursache genannt.
3. *Fehler im Datenfluss bzw. Objektzugriff*: Diese betreffen am häufigsten Datenflussanomalien.
4. *Schnittstellenfehler*: Bei diesen steht die die Nichtbeachtung von Kontrakten (d.h. die Verletzung von Vor- und Nachbedingungen bzw. die Nichtbeachtung von Seiteneffekten) im Vordergrund.
5. *Fehler im Kontrollfluss*: Diese betreffen am häufigsten die Behandlung von internen Ausnahmefällen (Exceptions) und parallelen Prozessen.
6. *Berechnungsfehler*: Die häufigste Fehler-Unterkategorie bezieht sich auf die Ressourcenverwendung von z.B. temporären Daten oder das Speicher-Management.
7. *Fehler in Klassen oder Datentypen*: Diese treten insgesamt am seltensten auf, betreffen dann aber am häufigsten die Typtransformation.

Bemerkenswert ist, dass Konfigurationsfehler an zweiter Stelle genannt wurden, noch vor den traditionellen Fehlerquellen wie Fehler im Datenfluss bzw. Objektzugriff und Schnittstellenfehlern. Fehler im Kontrollfluss und in der Berechnung, die insbesondere in der prozeduralen Entwicklung die größte Aufmerksamkeit durch Testmethoden erfahren haben, lagen in der Bedeutung stark zurück.

Die Definition der Fehlerkategorien, die Implementierung des Online-Fragebogens, die Benachrichtigung der Gewinner sowie die Auswertung der Daten haben wesentlich mehr Aufwand verursacht, als ursprünglich angenommen. Die große Anzahl der Teilnehmer und das Interesse innerhalb der Entwicklungsgemeinde hat dies entschädigt. So wurden die Ergebnisse auf Anfrage bei der iX-Konferenz „Bessere Software“, 22.-23. November 2005, in Köln präsentiert [Url:iX]. Zusätzliche Veröffentlichungen der Ergebnisse und eine weitergehende statistische Auswertung der Umfrage-Daten sind geplant.

Einladung: Wenn Sie die Ergebnisse der Umfrage mit Mitarbeitern des Arbeitskreises TOOP diskutieren bzw. eigene Erfahrungen, Kommentare oder Zusatzinformationen einbringen wollen, so laden wir Sie herzlich ein, über unsere Website [Url:TOOP] Kontakt mit uns aufzunehmen.

Danksagungen

Vielen Dank an den Verlag dpunkt für die Bereitstellung der Gewinne.

Dem OBJEKTspektrum, der Computer-Zeitung und heise.de danken wir für die Ankündigung der Umfrage, ohne die wir eine solch große Teilnehmerzahl nicht erreicht hätten.

Frau Barbara Paech (Leiterin der Arbeitsgruppe Software Systeme, Institut für Informatik, Ruprecht-Karls-Universität Heidelberg) danken wir für das Hosting des Fragebogens.

Folgende Mitglieder des Arbeitskreises TOOP (in alphabetischer Reihenfolge) haben neben den Autoren an der Umfrage mitgearbeitet: Michael Averstegge, Kay Krämer, Roger Müller, Mario Winter.

Wir danken auch den folgenden Kollegen für Ihre Reviews des Fragebogens: Michael Brunner, Jens Calame, Susanne Herl, Timea Illes und Christa Preisendanz.

Anhang A

Teilnahmebedingungen

Folgende Teilnahmebedingungen galten bei der Online-Umfrage:

Teilnehmer:

- Jede Person mit Erfahrung zu Software-Fehlern im Source-Code kann teilnehmen, wie z.B. Entwickler, Tester, Projektleiter und QS-Beauftragte.
- Jeder Teilnehmer darf nur einmal einsenden. Mehrfacheinsendungen einer Person werden von der Ziehung der Gewinne ausgeschlossen.

Ziehung der Gewinner:

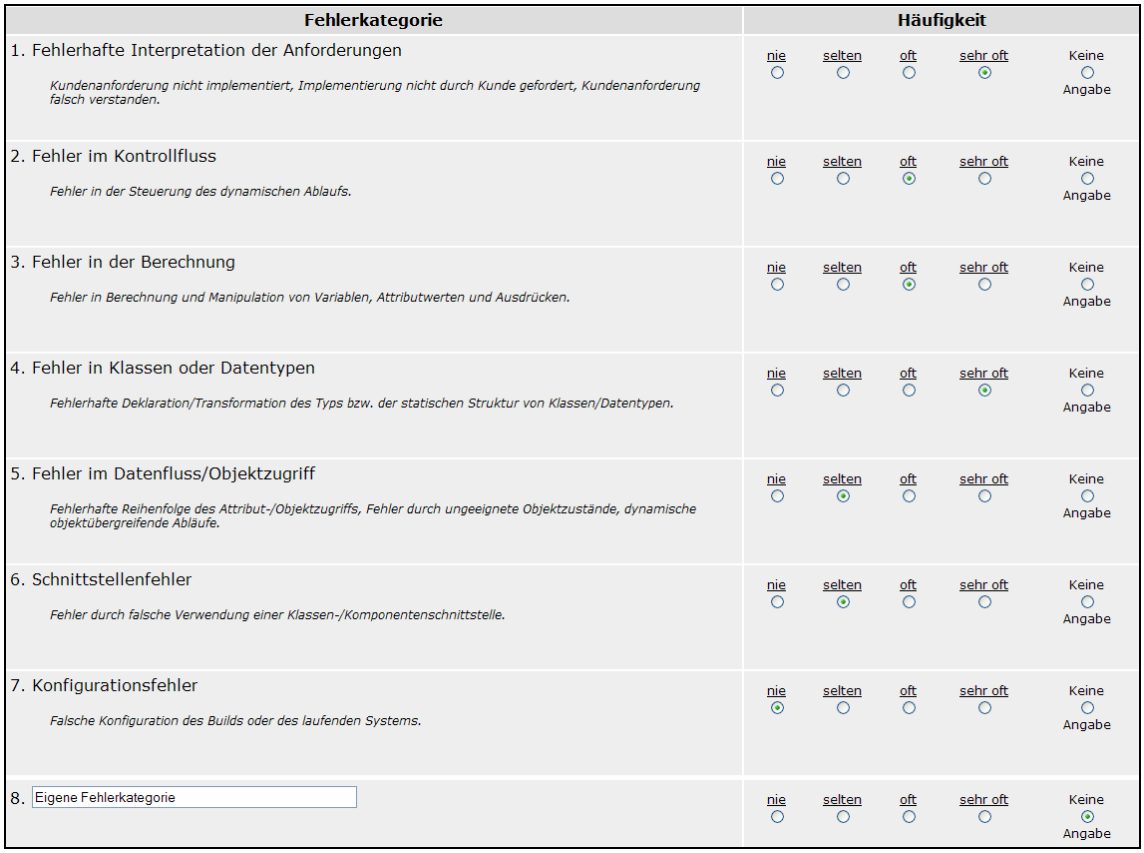
- Die Ziehung der Gewinner aus der Menge der Einsendungen zwischen offiziellem Start und Ende der Umfrage erfolgt öffentlich beim 23. TAV-Treffen am 17. und 18. November 2005 in München.
- Die Preise werden den Gewinnern im Anschluss an das Treffen per Post zugesandt.
- Die Gewinner werden mit Namen und Wohnort im Tagungsband des 23. TAV-Treffens sowie auf dieser Website bekannt gegeben.
- Der Rechtsweg ist ausgeschlossen.

Datenschutz:

- Die persönlichen Daten der Teilnehmer werden nur zum Zweck der Zusendung der Gewinne gespeichert. Nach erfolgter Ziehung werden diese Daten gelöscht.
- Die anonymisierten Daten zu den Fehlern werden zum Zweck der wissenschaftlichen Auswertung dauerhaft gespeichert.
- Die Gewinner werden mit Name und Wohnort auf der Webseite dieses Online-Fragebogens bekannt gegeben.

Anhang B

Screenshots



Fehlerkategorie	Häufigkeit				
1. Fehlerhafte Interpretation der Anforderungen *Kundenanforderung nicht implementiert, Implementierung nicht durch Kunde gefordert, Kundenanforderung falsch verstanden.*	nie ○	selten ○	oft ○	sehr oft ◉	Keine ○ Angabe
2. Fehler im Kontrollfluss *Fehler in der Steuerung des dynamischen Ablaufs.*	nie ○	selten ○	oft ◉	sehr oft ○	Keine ○ Angabe
3. Fehler in der Berechnung *Fehler in Berechnung und Manipulation von Variablen, Attributwerten und Ausdrücken.*	nie ○	selten ○	oft ◉	sehr oft ○	Keine ○ Angabe
4. Fehler in Klassen oder Datentypen *Fehlerhafte Deklaration/Transformation des Typs bzw. der statischen Struktur von Klassen/Datentypen.*	nie ○	selten ○	oft ○	sehr oft ◉	Keine ○ Angabe
5. Fehler im Datenfluss/Objektzugriff *Fehlerhafte Reihenfolge des Attribut-/Objektzugriffs, Fehler durch ungeeignete Objektzustände, dynamische objektübergreifende Abläufe.*	nie ○	selten ◉	oft ○	sehr oft ○	Keine ○ Angabe
6. Schnittstellenfehler *Fehler durch falsche Verwendung einer Klassen-/Komponentenschnittstelle.*	nie ○	selten ◉	oft ○	sehr oft ○	Keine ○ Angabe
7. Konfigurationsfehler *Falsche Konfiguration des Builds oder des laufenden Systems.*	nie ◉	selten ○	oft ○	sehr oft ○	Keine ○ Angabe
8. Eigene Fehlerkategorie	nie ○	selten ○	oft ○	sehr oft ○	Keine ◉ Angabe

Abbildung 23: Screenshot der ersten Fragebogenseite

Fehlerkategorie	Häufigkeit				
2. Fehler im Kontrollfluss					
2.1. Fehler in Prädikat *Fehlerhafte Logik in Verzweigungs- oder Schleifenend-Bedingung, die zu falschem Kontrollfluss führt. Beispiele: falscher boolscher Operator, falsche Klammerung von Operatoren.*	nie ○	selten ○	oft ○	sehr oft ○	Keine ○ Angabe
2.2. Fehler in Selektions-Konstrukt *Fehler in IF- oder Select-Konstrukt. Beispiele: fehlender Default-Fall, break vergessen, fehlende Blockbildung (geschweifte Klammern vergessen).*	nie ○	selten ○	oft ○	sehr oft ○	Keine ○ Angabe
2.3. Fehler in Schleifenkonstrukt *Beispiele: Fehlerhafter Initial- / Endwert bei FOR-Schleifen, falsche Inkrementierung, fehlerhafte Blockbildung.*	nie ○	selten ○	oft ○	sehr oft ○	Keine ○ Angabe
2.4. Fehler in Zustandslogik *Fehlerhafte oder fehlende Initialisierung des Objektzustands, fehlerhafter Zustandsübergang.*	nie ○	selten ○	oft ○	sehr oft ○	Keine ○ Angabe
2.5. Fehler in Behandlung von internem Ausnahmefall *Fehlerhafte Behandlung einer Exception, die von aufgerufener Komponente des eigenen Systems gemeldet wird. Beispiel: nicht behandelte Exception, finally vergessen.*	nie ○	selten ○	oft ○	sehr oft ○	Keine ○ Angabe
2.6. Methode auf falschem Objekt aufgerufen *Der Namen der aufgerufenen Methode ist richtig, aber der Aufruf geht nicht an das beabsichtigte Objekt. Beispiel: Klasse mit gleichem Namen aus falschem Paket aufgerufen.*	nie ○	selten ○	oft ○	sehr oft ○	Keine ○ Angabe
2.7. Falscher Methodenaufruf durch Polymorphismus bzw. Überladen *Der Name der aufgerufenen Methode ist richtig, aber der Aufruf geht an eine nicht-beabsichtigte Implementierung der Methode. Dies kann durch dynamische Bindung oder durch falsche Parametertypen bzw. -Anzahl verursacht werden.*	nie ○	selten ○	oft ○	sehr oft ○	Keine ○ Angabe
2.8. Falsche Methode ausgewählt *Mögliche Ursachen: irreführende Namensgebung der Methode; Name und Verantwortlichkeit der Methode nicht konsistent.*	nie ○	selten ○	oft ○	sehr oft ○	Keine ○ Angabe

Abbildung 24: Screenshot der zweiten Fragebogenseite

Anhang C

Literaturverzeichnis

[Beiz90] B. Beizer. *Software Testing Techniques*, 2. Ed. International Thomson Computer Press, 1990.

[Bind00] R. V.Binder. *Testing Object-Oriented Systems*. Addison-Wesley, 2000.

[Fent97] N. E. Fenton, S. Lawrence Pfleeger: *Software Metrics* 2. Ed. International Thomson Computer Press, 1997

[Link05] Johannes Link. *Softwaretests mit JUnit: Techniken der testgetriebenen Entwicklung*. 2., überarbeitete und erweiterte Auflage, dpunkt.verlag, Januar 2005, ISBN 3-89864-325-5.

[Myer04] G. Myers. *The Art of Software Testing*, 2. Ed. John Wiley & Sons, 2004.

[Rope94] M. Roper. Software Testing. McGraw-Hill, 1994, ISBN 0-07-707466-1.

[Spil05] A. Spillner, T. Linz: *Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester - Foundation Level nach ASQF- und ISTQB-Standard*. 3., überarbeitete Auflage, dpunkt.verlag, Juli 2005, ISBN 3-89864-358-1.

[Born06] L. Borner, M. Hamburg, S. Jungmayr. Fehlerhäufigkeiten in objektorientierten Systemen: Ergebnisse einer Online-Umfrage. Erscheint in *Softwaretechnik-Trends*, 2006.

[Thal02] Georg Erwin Thaller. *Software-Test: Verifikation und Validation*. 2., aktualisierte und erweiterte Auflage. Heise, April 2002, ISBN 3-88229-198-2.

[Url:CZ] http://www.computerzeitung.de

[Url:heise] http://www.heise.de/

[Url:iX] http://www.ix-konferenz.de/einstieg.php?konferenzid=5

[Url:OS] http://www.sigs-datacom.de/sd/publications/os/index.htm

[Url:Poster] http://toop.gi-ev.de/Ressourcen/iX_Poster_TOOP.pdf

[Url:TAV] http://www.gm.fh-koeln.de/~winter/tav/

[Url:TOOP] http://toop.gi-ev.de

[Url:Umfr] http://mars.informatik.uni-heidelberg.de:8080/Questionnaire_TooP/TooP/Questionnaire

[Vige04] U. Vigenschow. *Objektorientiertes Testen und Testautomatisierung in der Praxis: Konzepte, Techniken und Verfahren*. dpunkt.verlag, September 2004, ISBN 3-89864-305-0.

Dokument Information

Titel	Fehlerhäufigkeiten in objektorientierten Systemen: Basisauswertung einer Online-Umfrage
Datum	20.02.2006
Version	1.0
Status	freigegeben
Verteilung	http://www-swe.informatik.uni-heidelberg.de/research/publications/reports.htm