Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Guidelines als Hilfestellung bei checklistenbasierten Lesetechniken

Studienarbeit

im Studiengang Mathematik mit Studienrichtung Informatik

von

Donka Todorova

Prüfer: Prof. Dr. Kurt Schneider
Betreuer: Dipl.-Math. Thomas Flohr

Hannover, Dezember 2006

Erklärung

Hiermit versichere ich, dass ich die vorliegende Studienarbeit selbstständig und ohne fremde Hilfe verfasst habe und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe.

Hannover, 14.12.2006 Donka Todorova

Danksagung

Ich bedanke mich bei Herrn Dipl.-Math. Thomas Flohr für die hervorragende Betreuung meiner Studienarbeit.

Inhaltsverzeichnis

1. Einleitung

Erfahrungsgemäß treten in jedem Projekt während aller Phasen des Entwicklungsprozesses Fehler auf. Fehler können zu erheblichen Aufwänden für unproduktive Nachbearbeitungstätigkeiten führen, sofern sie nicht frühzeitig erkannt und behoben werden. Einen wichtigen Beitrag zur frühzeitigen Fehlererkennung leisten auf Projektebene Inspektionen. Der Inspektionsprozess gliedert sich in verschiedene Aktivitäten, wobei die individuelle Fehlersuche des Gutachters eine der wichtigsten ist. Dabei kommen verschiedene Lesetechniken in Einsatz. Die Lesetechniken unterstützen die Inspektoren beim effizienten Finden von Fehlern. Sie geben den Inspektoren Anleitungen **was** sie in dem Dokument prüfen sollen und **wie** sie die Inspektion durchführen sollten.

Fehlersuche mit checklistenbasiertem Lesen (*engl.: Checklist-Based Reading, CBR*) ist die am weitesten verbreitete Lesetechnik. Diese Methode verwendet, wie der Name schon sagt, eine Checkliste. Die Checkliste besteht am meisten aus Fragen, die der Gutachter beim Ablesen eines Softwaredokumentes beantworten muss. Die meisten Fragen in den Checklisten sind Entscheidungsfragen. Entscheidungsfragen sind die Fragen, auf die man nur mit „Ja“ oder mit „Nein“ antworten kann. Diese Fragen geben dem Gutachter Hinweise darauf, “**was**” er in einem Dokument inspizieren soll. CBR ist eine nicht-systematische Lesetechnik, weil es Guidelines (Handlungsanleitungen) fehlen, “**wie**” man die Inspektion durchführen kann, d.h. wie ein Checkpunkt überprüft werden muss, bzw. wann er erfüllt ist.

Diese Studienarbeit soll eine Übersicht über die verschiedenen Lesetechniken geben. Ein wichtiger Aspekt dabei ist die Untersuchung ob und wie Guidelines in den verschiedenen Lesetechniken integriert sind. Ziel der Arbeit ist die Erarbeitung von mehreren Modellen zur Integration von Guidelines bei checklistenbasierten Lesentechniken. Um Antwort auf die Fragen wie: „ Ist die Integration von Guidelines in Checklisten sinnvoll?“ und „Welches Modell mit Guidelines unterstützt die Gutachter am meisten?“ zu finden, werden zuletzt die Ergebnisse der durchgeführten Umfrage mit Studenten der Leibniz Universität Hannover beschrieben.

1.1. Aufbau der Arbeit

In Kapitel 2 wird zunächst die Inspektionstechnik kurz vorgestellt. Dabei werden die Ziele und die qualitative Vorteile der Inspektion dargestellt. Es wird den typischen Inspektionsprozess und die verschiedenen Lesetechniken beschrieben und zuletzt werden die Charakteristiken von den Inspektionslesetechniken aufgezählt.

Im dritten Teil der Arbeit wird zuerst der Begriff „Guideline“ definiert. Danach wird

darüber diskutiert, in welchen Lesetechniken die Guidelines integriert sind und welchen Einfluss sie auf den Inspektionsprozess haben. Anschließend werden die verschiedenen Formulierungsformen der Guidelines vorgestellt.

Im Kapitel 4 sind die verschiedenen Modelle der Checklisten mit integrierten Guidelines grafisch und textuell dargestellt. Dies erfolgt anhand Beispiele. Außerdem werden die Vorteile und die Nachteile der vorgestellten Modelle beschrieben.

Im nächsten Kapitel wird die durchgeführte Umfrage beschrieben und die Auswertung mit Unterstützung von Diagrammen gegeben.

Das sechste und letzte Kapitel schließt mit einer Zusammenfassung und einem kurzen Ausblick.

2. Inspektion

Eine Inspektion ist eine Analysetechnik zur Überprüfung und Beurteilung von Software-Produkten. Das Ziel der Inspektionen ist das Finden von Fehlern, Verletzungen von Entwicklungsstandards oder Mängeln des Produkts, um die Software Qualität zu verbessern und die Produktivität des Programmierers zu steigern. Inspektionen werden in allen Phasen der Software-Entwicklung eingesetzt, wobei Inspektionen für Kode und Entwurf am üblichsten sind. Im Vergleich zum dynamischen Verfahren, wie zum Beispiel dem Testen, können Inspektionen aber schon sehr viel früher in der Entwicklung eingesetzt werden. Dadurch werden die Qualitätsdefizite dort gefunden und beseitigt, wo sie auch tatsächlich entstehen. Inspektionen wurden erstmalig von Fagan 1974 bei IBM eingeführt [10].

Die folgende Graphik gibt eine Übersicht über die wesentlichen Elemente einer Inspektion:

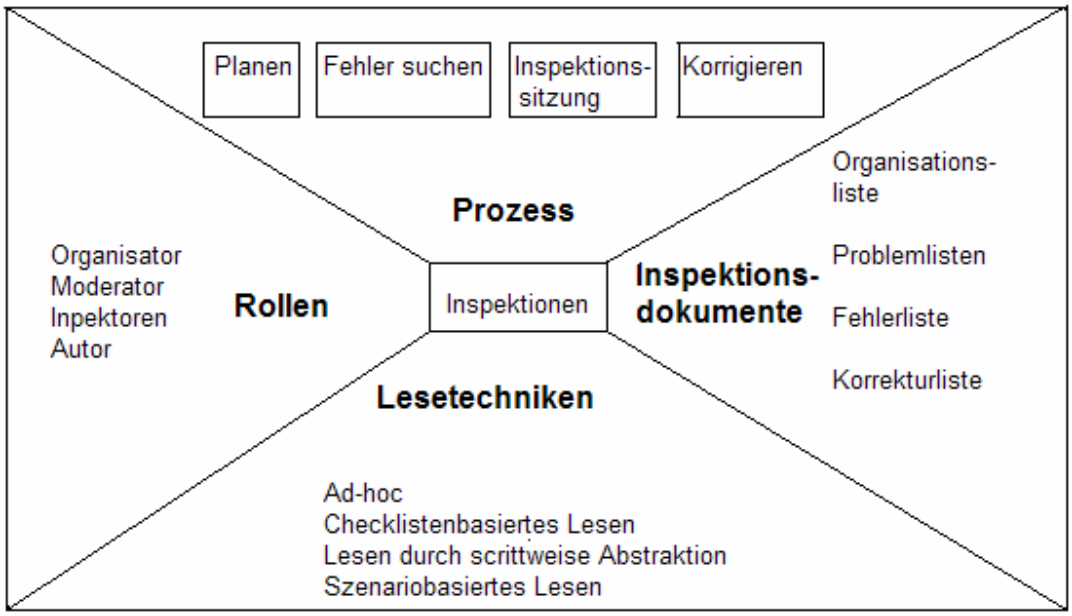


Abbildung 1 aus [16]

2.1. Ziele und qualitative Vorteile einer Inspektion

Die Ziele einer Inspektion sind:

- Sichtbare Fehlerreduktion zu erzielen,
- Qualitätsverbesserung durch Fehlereliminierung,
- Eine deutliche Kosteneinsparung zu erreichen,
- Die Wiederverwendbarkeit des Softwareproduktes zu überprüfen.

Die qualitativen Vorteile von Inspektionen sind vielfältig. Die Einsetzung von Inspektionen aller Art führt zu [29]:

- Der frühen Entdeckung von Fehlern,
- Vereinfachung der Programme,
- Vereinheitlichung des Programmierstils,
- Einer transparenten Software Entwicklung,
- Zuverlässigerer Termin- und Aufwandsschätzung,
- Erhöhter Lerneffekt,
- Verbesserung der Dokumentation und Einhaltung von Standards,
- Erhöhter Zufriedenheit des Benutzers.

2.2. Der Inspektionsprozess

Der typische Inspektionsprozess besteht aus vier Phasen - Planen, Fehler suchen, Sammeln und Entscheiden (Inspektionssitzung) und Korrigieren (siehe Abbildung 1).

Planen - das Planen wird durch die Rolle Organisator gemacht und dokumentiert in einer Organisationsliste. In der Planung wird die Inspektion organisiert. Es wird ein Zeit-Aufwand und Ressourcenplan erstellt. Die Organisation beinhaltet die Aufstellung des Inspektionsteams, die Rollenaufteilung, die Terminplanung einer oder mehreren Inspektionssitzungen und die Auf- und Verteilung des Inspektionsmaterials.

Fehler suchen - In dieser Phase wird das Dokument auf alle möglichen Fehler gründlich untersucht. Das Dokument wird von einem oder mehreren Inspektoren mit der Unterstützung einer Lesetechnik gelesen und dokumentiert in Problemlisten. Jeder Inspektor notiert jeden Fehler und jede Unklarheit. Dabei ist ein Mangel nicht unbedingt einem Fehler gleichzusetzen. Der vom Gutachter entdeckte Mangel erfordert jedoch eine Klärung in der Inspektionssitzung.

Sammeln und Entscheiden (Inspektionssitzung) - Die Inspektionssitzung ist der wichtigste Teil einer Inspektion. An der Inspektionssitzung nehmen in der Regel der/die Inspektoren, der Autor des Dokuments und ein Moderator teil. Der Moderator übernimmt die Leitung der Inspektionssitzung. In dieser Phase wird darüber entschieden, ob ein Fehler wirklich als Fehler oder nur als Mangel angegeben werden kann. Es wird in einer Fehlerliste bzw. einem Sitzungsprotokoll dokumentiert.

Korrigieren - Die im Inspektionsprotokoll dokumentierten Fehler werden vom Autor korrigiert und die Ergebnisse werden in einer Korrekturliste dokumentiert.

Ziel des Prozesses ist es, aus dem Eingangsdokument Softwareprodukt das Aus-

gangsdokument korrigiertes Softwareprodukt inklusiver der Informationen über den Inspektionsprozess zu erhalten. Dieser Inspektionsprozess kann für jede Phase der Entwicklung der Software durchgeführt werden.

Das nächste Bild zeigt den typischen Inspektionsprozess, der aus vier Phasen besteht:

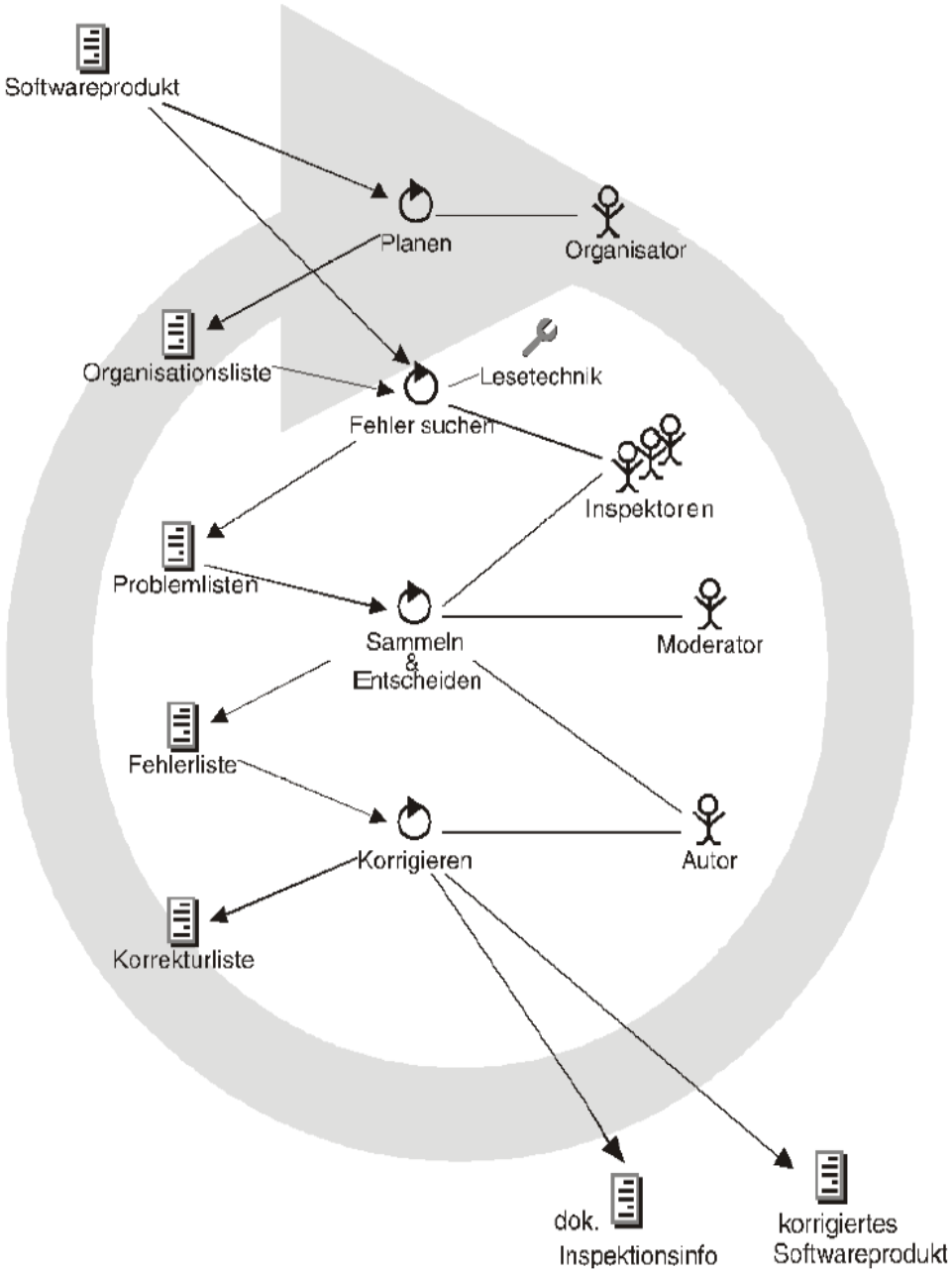


Abbildung 2 aus [26]

2.3. Lesetechniken

In diesem Abschnitt werden zunächst die verschiedenen Lesetechniken vorgestellt. Lesetechniken sind Hilfsmittel für die Inspektoren, mit deren Hilfe sich die Inspektoren auf die Inspektionssitzung vorbereiten. Sie bezeichnen Strategien für die Fehlersuche in Softwaredokumenten.

Die folgenden Lesetechniken sind zu unterscheiden:

Ad-hoc

Checklistenbasiertes Lesen (CBR)

Lesen durch schrittweise Abstraktion

Szenarienbasiertes Lesen
- Fehlerklassenbasiertes Lesen (DBR)
- Perspektivenbasiertes Lesen (PBR)
- Tracebility-basiertes Lesen
- Usage-basiertes Lesen

2.3.1. Ad-hoc Lesen

Das Ad-hoc Lesen, wie der Name schon sagt, stellt keine Anleitungen zur Verfügung, wie der Inspekor bei der Überprüfung vorgehen muss und worauf er zu achten hat. Folglich müssen die Inspekoren auf ihre eigene Intuition und Erfahrung zurückgreifen, um festzustellen, wie man am besten die Fehler in einem Softwaredokument findet. Die Ad-Hoc Vorgehensweise hat zur Folge, dass für Dritte nicht nachvollziehbar ist, wie die Ergebnisse zustande kamen. Weiterhin ist es für einen Ispektor schwer einschätzbar, wie viele der im Dokument enthaltenen Fehler er gefunden hat.

2.3.2. Checklistenbasiertes Lesen (CBR)

Fehlersuche mit checklistenbasiertem Lesen (*engl*.: *Checklist-Based Reading, CBR*) ist die einfachste Lesetechnik nach der Ad-hoc Fehlersuche. Diese Methode verwendet eine Checkliste. Anhand dieser Checkliste gehen die Reviewer schrittweise vor und bearbeiten jeden Punkt bzw. jede Frage auf der Checkliste. Die Fragen geben dem Gutachter Hinweise darauf, „was“ er in einem Dokument inspizieren soll. Heute wird CBR als Standardlesetechnik in den Softwareorganisationen betrachtet [17]. Folglich wird CBR häufig als Grundmethode in den empirischen Studien verwendet, wenn Lesetechniken nachgeforscht werden. Siehe Tabelle 1 und 2.

TABELLE 1
Summery of Studies Investigating Defect-Based Reading (DBR)

Study	Purpose	Environment	Subjects	Significant?
Porter et al. [21] -1995	DBR vs. Ad-hoc and CBR	Academic	24+24	YES
Fusaro et el.[9] -1997	DBR vs. Ad-hoc and CBR	Academic	30	NO
Miller et al.[20] -1998	DBR vs. CBR	Academic	50	Inconclusive
Sandahl et al.[25]-1998	DBR vs. CBR	Academic	24	NO
Porter et al.[22] -1998	DBR vs. Ad-hoc and CBR	Industrial	18	YES

Tabelle 1 aus [30]

TABELLE 2
Summery of Studies Investigating Perspective-Based Reading for Requirements Documents

Study	Purpose	Environment	Subjects	Significan?
Basili et al.[2]-1996	PBR vs. Ad-hoc	Industrial	12+13	YES
Ciolkowski et al.[5]-1997	PBR vs. Ad-hoc	Academic	25+26	YES
Shull[27]-1998	PBR vs. Ad-hoc	Academic	66	YES
Regnell et al.[23]-2000	Are different faults detected by different perspectives in PBR?	Academic	30	NO
Lanubile and Visag-gio[13]-2000	PBR vs. Ad-hoc and CBR	Academic	114+109	NO
Biffl [4]-2001	PBR vs. CBR	Academic	169	NO
Halling [12]-2001	PBR vs. CBR	Academic	177	NO

Tabelle 2 aus [30]

2.3.3. Lesen durch schrittweise Abstraktion

Die erste Idee für diese Lesetechnik wurde von Harlan D. Mills 1972 vorgeschlagen [20]. Er hat eine Lesetechnik entwickelt, um Fehler in Kodedokumenten zu finden, die „ Lesen durch schrittweise Abstraktion “ genannt wurde. Der Gutachter identifiziert elementare Kodebausteine, bestimmt die Funktion der Kodebausteine und fasst

dann diese Kodebausteine zu größeren Bausteinen zusammen. Dann ist es möglich, die abgeleitete Abstraktion mit der Spezifikation zu vergleichen, um Fehler zu finden. Die Effektivität und die Effizienz dieser Technik wurden von Basili und Selby 1987 validiert [1].Sie haben gezeigt, dass durch das Lesen durch schrittweise Abstraktion mehrere Fehler gefunden werden können, als durch funktionelles oder strukturelles Testen.

2.3.4 Szenarienbasiertes Lesen

Das Szenarienbasierte Lesen definiert sowohl wie die Inspektoren vorgehen sollen als auch worauf diese zu achten haben. Dazu erhalten die Inspektoren konkrete Anleitungen (so genannte Szenarien). Es existieren fehlerklassenbasiertes Lesen, perspektivenbasiertes Lesen, Tracebility-basiertes Lesen und Usage-basiertes Lesen.

2.3.4.1. Fehlerklassenbasiertes Lesen (DBR)

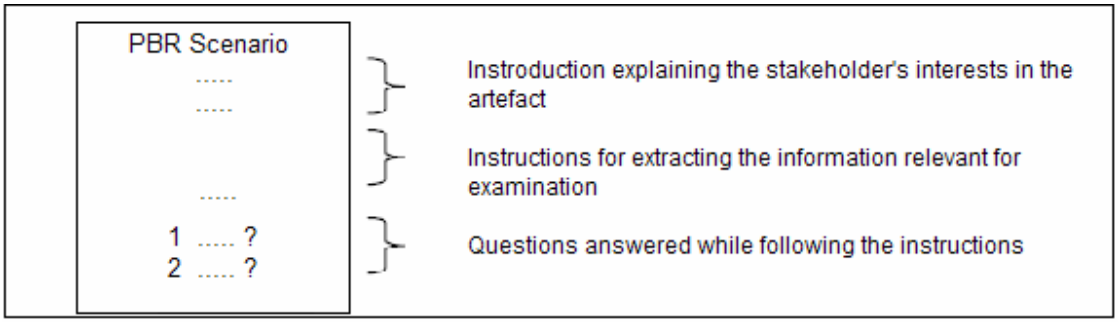
Fehlerklassenbasiertes Lesen ist eine systematische Lesetechnik. [19]. Diese Technik wurde ursprünglich entwickelt, um Fehler in Anforderungsdokumenten aufzudecken. Für diese Technik werden Fehler in Gruppen aufgeteilt und für jede Fehlergruppe werden genaue Fragestellungen und, im Unterschied zum checklistenbasierten Lesen, Vorgangsweisen bei der Fehlersuche entwickelt. Die Inspektoren beantworten die Fragestellungen, indem sie das vordefinierte Szenario durchgehen. Mit einem Szenario kann man meist nicht alle Fehlerklasse finden, deswegen sind mehrere Sichtweisen oder Szenarien oft notwendig.

2.3.4.2. Perspektivenbasiertes Lesen (PBR)

Perspektivenbasiertes Lesen (*engl.: perspective-based reading, PBR*) ist eine szenariobasierte Lesetechnik, die den Blickwinkel der Konsumenten eines Softwareprodukts verwendet, z.B. Kunde, Designer oder Tester für ein Anforderungsdokument. Für jede Konsumentenperspektive gibt es ein Szenario, das den Inspektor anleitet, ein entsprechendes Modell zu produzieren. Ein Szenario leitet einen Inspektor durch ein Dokument mit einer bestimmten Rolle aus einer Menge von Rollen, die kombiniert das Dokument komplett abdecken sollen.

Prinzipiell ergeben sich in der perspektivenbasierten Lesetechnik bei der Vorgehensweise 3 Schritte:

- Erklären der jeweiligen Interessen an dem Reviewobjekt (jeweils aus Kunden-, Designer-, oder Testersicht)
- Spezielle Anweisungen für die Inspektoren, die er bei der Fehlersuche berücksichtigen muss
- Eine Liste von Fragen, die während der Fehlersuche beantwortet werden müssen



Inhalt und Struktur eines PBR Szenarios
Abbildung 3 aus [18]

2.3.4.3. Tracebility-basiertes Lesen

Tracebility-basiertes Lesen ist eine Lesetechnik, die das Lesen von objekt-orientierten Designdokumenten unterstützt. Diese Methode wird in vertikales und horizontales Lesen geteilt. Das vertikale Lesen soll prüfen, ob das Design zu den Anforderungen korrespondiert. Das horizontale Lesen prüft die Design-Artefakte gegeneinander (z.B. Klassendiagramme, Paketdiagramme, usw.).[30]

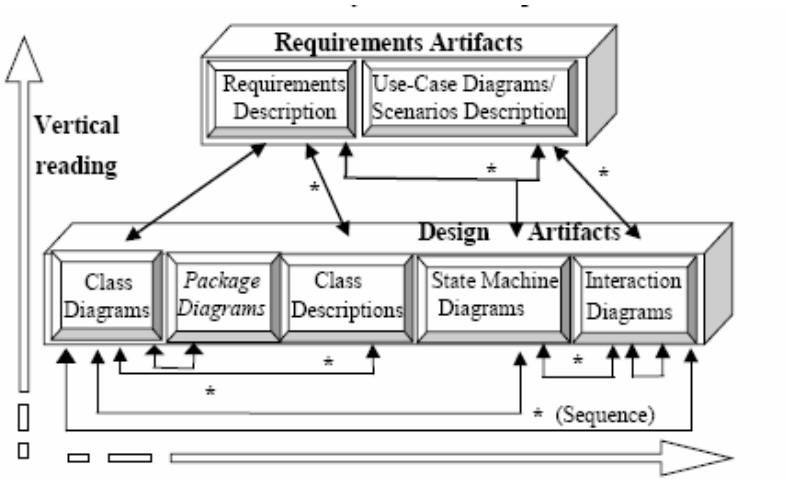


Abbildung 4 aus [32]

2.3.4.4. Usage-basiertes Lesen

Die meisten anderen Lesetechniken zielen darauf ab, möglichst viele Fehler zu finden, unabhängig von ihrer Wichtigkeit. Bei Usage-basiertem Lesen (UBR) konzentriert sich der Inspektor auf die schweren Fehler, die das Endprodukt des Reviewobjektes aus Benutzersicht am stärksten negativ beeinflussen. Diese Lesetechnik kann für jede Art von Dokumenten eingesetzt werden. Dabei werden dem Gutachter Use-Cases zur Verfügung gestellt.

Die Grundschritte dieser Methode sind wie folgt definiert (siehe Abbildung 5):

1. Prioritize the use cases in order of importance from a user perspective.
In the preparation phase of inspections,
2. Select the use case with the highest priority.
3. Track the use case's scenarios through the document under inspection.
4. During tracking, ensure that the document under inspection fulfils the use case's goal; for example, make sure it provides the needed functionality and that the interfaces are correct. Identify and report the issues that tracking reveals.
5. Select the next use case and repeat from step 3 until the time is up or you've covered all use cases. [31]

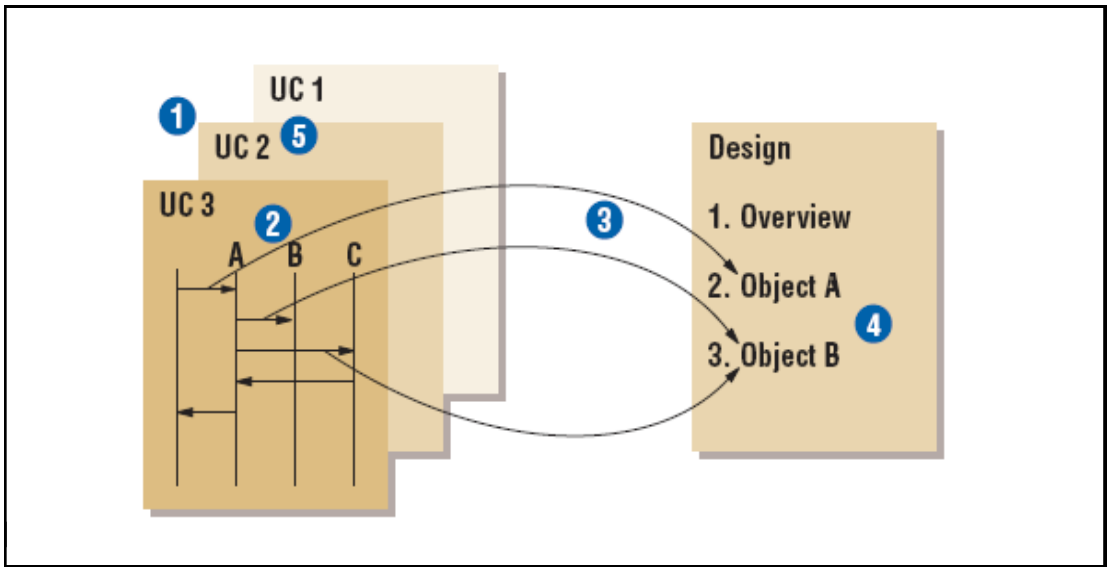


Abbildung 5. Usage-based reading. The letters in the use case document correspond to those in the design document. So, when something is happening with object A in the use case, the reviewer checks it in the corresponding section in the design document. For example, if A is a taxi and B a central unit, there will be communication signals between A and B. [31]

Es besteht eine gewisse Ähnlichkeit zur Kundensicht im PBR, da diese im Normalfall auch mit Use Cases arbeitet. Meistens werden jedoch im Zuge der Fehlersuche im PBR die Use Cases erst entwickelt, im UBR sollten sie aber schon vorab definiert sein. Im Unterschied zu PBR ist UBR nicht für unterschiedliche Szenarios der gleichen Kategorie (z.B. Anforderungsdokumente) zu verwenden. Das Szenario muss in UBR für jedes Projekt neu entwickelt werden, weil die Use Cases schon vorher feststehen.

2.4. Charakteristika von Inspektionslesetechniken

In diesem Abschnitt werden die verschiedenen Kriterien vorgestellt, mit denen eine Lesetechnik klassifiziert und beurteilt werden kann. Die Kriterien sollen Antworten zu den folgenden Fragen zur Verfügung stellen:

1. Anwendungsspektrum (engl.: Application Contex): Auf welche Softwareprodukten kann eine Lesetechnik angewendet werden und auf welchen Softwareprodukten ist eine Lesetechnik bereits angewendet worden?

2. Anleitung (engl.: Usability): Stellt die Lesetechnik Guidelines zur Verfügung, wie ein Inspektor die Fehlersuche durchführen kann?

3. Wiederholbarkeit (engl.: Repeatability): Sind die Ergebnisse von der Arbeit eines Inspektor wiederholbar, d.h. lassen sich die Ergebnisse unabhängig von der Person reproduzieren?

4. Anpassbarkeit (engl.: Adaptability): Ist die Technik an verschiedene Produkte anpassbar?

5. Überdeckung (engl.: Coverage): Deckt die Lesetechnik weitgehend das Auffinden aller Fehler im Produkt ab?

6. Überlappung (engl.: Overlap): Vermeidet die Technik die Überlappung zwischen Inspektoren?

7. Validation: Wie breit ist die Lesetechnik bis jetzt angewendet worden?

Die untenstehende Tabelle charakterisiert jede Lesetechnik entsprechend diesen Kriterien. Es werden Fragezeichen für diejenige Fälle verwendet, für die es keine klare Antwort gibt.

	Characteristic						
	Application Context	Usability	Repeatability	Adaptability	Coverage	Overlap	Validation
Ad-hoc	All Products: All Products	No	No	No	Low	Hight	Industial Practice
Checklists	All Products: All Products	No	No	Yes	Case dependent	Hight	Industial Practice
Reading by stepwise Abstraction	All Products allowing abstaction; Functional Code	Yes	Yes	No	Hight	Hight	Applied in Cleanroom projects
Defect-based reading	All Products; Requirements	Yes	Case dependent	Yes	Hight	?	Experimental Validation
Perspective-based reading	All Products; Requiremants, Code	Yes	Yes	Yes	Hight	?	Experimental Validation

Tabelle 3 aus [13] *Characterization of Reading Techniques*

3. Guidelines und die Lesetechniken

3.1. Eigene Definition des Begriffs „Guideline“

Guidelines sind Anleitungen in einem Inspektionsprozess, die dem Gutachter helfen zu wissen, wie er die Fehlersuche durchführen kann. Sie geben den Inspektoren Hinweise, wie sie die richtige Entscheidung treffen könnten, ob ein Checkpunkt richtig ausgeführt ist und sollen sicherstellen, dass ein Checkpunkt von allen Gutachtern gleichbedeutend interpretiert wird. Guidelines helfen ein besseres Verständnis für das Dokument zu gewinnen und die relevante Information für die Prüfung zu sammeln. Ihrem Einsatz entsprechend, werden die Guidelines unterschiedlich formuliert.

3.2. Guidelines in den verschiedenen Lesetechniken

Aus Tabelle 3 könnte es abgelesen werden, dass für die Anwendung der Techniken durch den Gutachter außer für Ad-Hoc-Lesen und Checklistenbasierten Lesen Guidelines gibt (weiter im Text werden die Begriffe “Guidelines“ und “Anleitung“ als Synonyme verwendet).

Bei der Ad-Hoc Methode wird keine Richtung dafür vorgegeben, wie der Gutachter seine Inspektion durchführt und worauf er achten sollte.

Das checklistenbasierte Lesen gibt durch die Checkpunkte Hinweise darauf, was der Gutachter inspizieren sollte, verzichtet aber auch auf Guidelines, wie die Inspektion durchgeführt werden könnte.

Das Lesen durch schrittweise Abstaktion gibt dem Gutachter Guidelines, wie er beim Lesen vorzugehen hat. Es wird ausführlich beschrieben, welche Grundaktivitäten bearbeitet werden können:

- wie das Programm in Unterprogrammmodule aufgeteilt wird,
- wie diese Untermodule beschreiben werden müssen,
- wie die Untermodule in ein Ganzes zusammengefügt werden,
- und wie diese oben aufgeführte Arbeitsschritte mit den ursprünglichen Spezifikationen zu vergleichen sind.

Das szenarienbasierte Lesen enthält detaillierte Guidelines, die ein Leser ausführen soll („active guidance“). Diese Anleitungen ermutigen die Inspektoren aktiv mit dem Dokument zu arbeiten z.B. Notizen zu machen, das Dokument zu kommentieren oder ein kleines Modell zu bauen.

In der Literatur sind am meisten Szenarien für Perspektivenbasiertes Lesen zu finden.

Im Anschluss wird ein Teil eines Szenarios für Perspektivenbasiertes Lesen mit seinen wichtigsten Phasen dargestellt (das komplette Szenario ist im Anhang dieser Arbeit zu finden).

Beispiel

Integrationstest-Szenario
Lesen aus der Sicht eines Integrationstesters

Entwickeln Sie aus der Spezifikation bzw. aus dem Entwurf Testfälle, mit denen Sie die korrekte Funktionsweise des Modules für die Ein- und Ausgabeobjekte überprüfen können.

Berücksichtigen Sie insbesondere die physikalischen Ein- und Ausgabegrößen. Stellen Sie sich dabei vor, dass das Modul in einem bereits vorhandenen System von Modulen integriert wird. Beachten Sie dabei, dass Sie alle möglichen Eingabekombinationen durch je einen Testfall berücksichtigen. Benutzen Sie dabei nicht nur die Testfälle, die auf der Spezifikation bzw. dem Entwurf vorgegeben sind. Schreiben Sie die Testfälle auf das Formblatt „Ergebnisse Integrationstest-Szenario“ auf.

Extrahieren Sie den Kontrollflussgraphen aus dem Kodedokument. Fassen Sie dabei geeignete Kodesequenzen (z.B. eine Folge von Anweisungen) zusammen. Achten Sie hierbei besonders darauf, dass im Kontrollflussgraphen alle Eingabeobjekte in das Modul und alle Ausgabeobjekte aus dem Modul identifiziert werden können. Benutzen Sie bei der Erstellung des Kontrollflussgraphen die graphischen Symbole des Formblatts „Symbole für Test-Szenarien“.

Schreiben Sie den Kontrollflussgraphen auf das Formblatt „Ergebnisse Integrationstest-Szenario“ und markieren Sie Ein- bzw. Ausgabeobjekte und Ein- bzw. Ausgabeanweisungen im Kontrollflussgraphen.

Benutzen Sie die Testfälle, die Sie für die Eingabeobjekte aufgestellt haben, als Input in das Modul. Stellen Sie mit Hilfe Ihres Kontrollflussgraphen fest, ob das durch den Testfall erzeugte Verhalten der von Ihnen identifizierten Ausgabeobjekte mit dem in der Spezifikation bzw. dem Entwurf definierten Verhalten übereinstimmt. Falls Unterschiede vorhanden sind, überprüfen Sie ob ein Mangel vorliegt oder nicht.
Beispiel aus[15]

3.3. Einfluss der Guidelines auf die Lesetechniken

Nach Gilb[11] der wesentliche Punkt jeder Inspektion ist die Fehlerentdeckungsphase, wo die Inspektoren versuchen möglichst viele Fehler zu identifizieren. Das Lesen stellt die Grundaktivität bei dem Finden von Fehlern in Softwaredokumenten dar. Deswegen wurden mehrere Experimenten durchgeführt, um die verschiedenen Lesetechniken in Bezug auf Effektivität und Effizienz zu vergleichen. Es wurden

auch einige Untersuchungen gemacht, um festzustellen welchen Einfluss Guidelines auf die Lesetechniken haben. Sie hatten zum Ziel herauszufinden, ob Guidelines den Inspektionsprozess verbessern könnten.

Winkler, Biffl und Thurnher haben die Auswirkung des Guidelines bei Entwurf- Inspektionen untersucht [33]. Sie haben traditionelle Checklisten, Checklisten mit integrierten Guidelines und Usage-basiertes Lesen verglichen. Diese Studie hat gezeigt: (a) Checklisten mit Guidelines sind deutlich effektiver als traditionelle Checklisten für das Finden von major Fehlern (b) Guidelines verbessern die Effektivität und die Effizienz des Inspektionsprozesses und (c) Usage-basiertes Lesen ist effektiver und effizienter als die beiden oben genannten Typen von Checklisten.

Mit Studenten der Universität Bari wurde ein anderes Experiment durchgeführt [14]. Sie haben szenarienbasiertes Lesen (mit Guidelines) und „fokussierte" Checklisten (ohne Guidelines) verglichen. Die beiden Lesetechniken wurden so konstruiert, dass sie dieselben Perspektiven und Aspekten erfassten. Der einzige Unterschied dabei war, dass bei dem szenarienbasierten Lesen Guidelines eingesetzt worden waren. Die Resultate zeigten statistisch unbedeutende Unterschiede ($P < 0,05$) zwischen den beiden Lesetechniken. Jedoch vorwiesen die Prüfer eine bessere Annahme und intensiveres Interesse an den „fokussierten" Checklisten als Leseszenarien.

In einem anderen Experiment wurde nochmals untersucht, ob Guidelines den Inspektionsprozess verbessern [6]. Um die Auswirkung der Guidelines zu analysieren, wurden die Resultate von zwei Inspektionsteams verglichen. Das erste Team hat PBR (mit Guidelines) verwendet und das zweite Team hat CBR (ohne Guidelines) verwendet. Detailliert wurden die folgenden zwei Hypothesen analysiert:
Hypothese 1- *Team Effektivität:* Inspektionsteams finden mehr Fehler mit Guidelines als ohne Guidelines.
Hypothese 2- *Team Effizienz:* Inspektionsteams finden mehr Fehler pro Zeitintervall mit Guidelines als ohne Guidelines.
Die Ergebnisse dieser Studie beweisen, dass Guidelines die Effektivität des Inspektionsprozesses verbessern könnten, aber negativen Einfluss auf die Effizienz des Prozesses haben. Ähnliche Ergebnisse sind auch in [7] beschrieben.

Die Ergebnisse der oben genannten Studien stellen eine gewisse Tendenz dar, aber sind nicht endgültig. Allgemein ist festzustellen, dass Guidelines zu einem besseren Verständnis des Dokumentes und zu einer strukturierten Arbeit bei der Fehlersuche führen. Als Folge davon verbessern sich die Effektivität und die Effizienz des Inspektionsprozesses. Der Nachteil dieser Studien ist, dass sie mit einer kleinen Anzahl von Testpersonen durchgeführt worden sind. Deswegen können die Ergebnisse nur vorläufig sein. Zukünftige Untersuchungen sind empfehlenswert.

3.4. Formulierungsformen der Guidelines

3.4.1. Frageform

Die Guidelines können als Fragen formuliert werden. In diesem Fall werden zu einem Checkpunkt eine oder mehrere Erweiterungsfragen gestellt. Das Ziel der Fragen ist zu betonen, welche Aspekte bei der Checkpunküberprüfung berücksichtigt werden müssten. Zum Beispiel der Checkpunkt „ Use Cases sind vorhanden?" kann mit der Folgefrage " Sind alle Elemente in den Use Cases beschrieben?" erweitert werden. Es ist wichtig, dass die Fragen einfach, klar und direkt formuliert sind.

3.4.2. Allgemeine Handlungsform

Eine andere Möglichkeit der Formulierung ist die Handlungsform. Die Guidelines in diesem Fall leiten den Gutachtern an, was sie tun sollten, um den Checkpunkt genauer zu prüfen. Zum Beispiel können sie so formuliert werden: **„ Achten Sie darauf, dass..“**, **„Berücksichtigen Sie..“** oder **„Überprüfen Sie..“**. Durch die Guidelines können die Inspektoren relevante Information für die Prüfung des entsprechenden Checkpunktes sammeln.

3.4.3. Schrittweise Handlungsform

Die meistbenutzte Form für die Formulierung des Guidelines bei PBR ist die Schrittweisebeschreibung. Diese soll die Gutachter bei konstruktiver Fehlersuche unterstützen, indem sie die Aktivitäten schrittweise beschreibt. Die schrittweise Handlungsform kann auch beim checklistenbasierten Lesen eingesetzt werden. Die Instruktionen geben konkrete Arbeitsaufgaben vor, mit deren Hilfe die Checkpunkte geprüft werden könnten. Diese Formulierungsform ist sehr hilfreich, vor allem für unerfahrene Inspektoren. Ein Nachteil ist, dass die Guidelines sehr lang sein können und dadurch würde sich die komplette Form der Checklisten verändern. Eine solche Beschreibung kann den folgenden Aufbau haben:

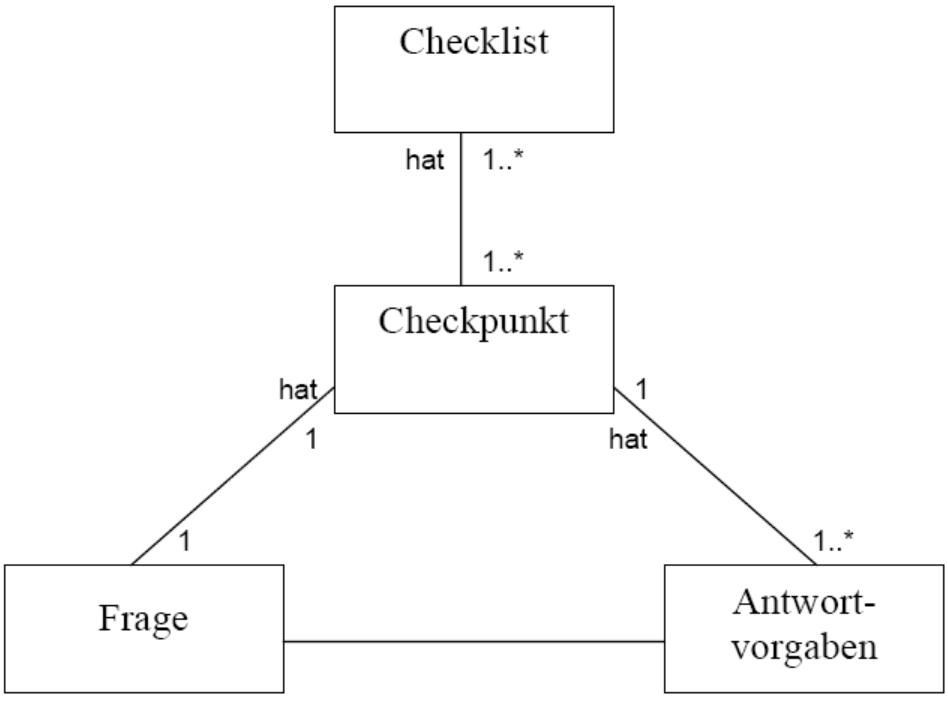
No.	Schritt
1	Beschreibung des aktuellen Schritts und Darstellung des benötigten
	Template
2	.....

Abbildung 6: *Schrittweise Handlungsform*

4. Modelle von Checklisten mit integrierten Guidelines

4.1. Allgemeine Beschreibung einer Checkliste

Allgemein besteht eine Checkliste aus Checkpunkten. Ein Checkpunkt besteht aus einer Frage und der dazugehörigen Antwortvorgabe. In [28] sind ausführlich die verschiedenen Modelle Checkliste beschrieben. Diese unterscheiden sich in der Ausformulierungsform der Fragen und in dem Variablentypen, der als Antwort zurückgegeben wird.



Checklist allgemein

Abbildung 7 aus [28]

Das einfachste und häufig benutzte Modell ist das „Kästchen“ Modell. Die Checkpunkte dieses Modells sind die Entscheidungsfragen, auf die man nur mit “Ja“ oder “Nein” antworten kann. Als Antwortvorgabe ist nur ein leeres Kästchen gegeben. In

diesem Fall wird die Frage abgehackt, wenn die beschriebene Anweisung korrekt ausgeführt ist.

Als Beispiel dafür wird ein Teil von Checklisten Modell „Kästchen“ dargestellt.

Die Entwurfsspezifikation verwendet das verteilte Template.	☐
Wichtige übergreifende Konzepte und Entwurfsentscheidungen sind beschrieben, erläutert und begründet.	☐
Wichtige Schnittstellen zur Außenwelt sind beschrieben. Jede Schnittstelle, die auch noch andere (Projekte) betreffen könnte, ist exakt (z.B. formal) beschrieben.	☐
Grundkonzepte der Funktionsverteilung auf verschiedene Rechner oder Systemteile sind beschrieben und begründet. Die Kommunikation zwischen diesen Teilen ist beschrieben.	☐
Es gibt eine grafische Übersicht der verwendeten Packages/Module. Die Packages/Module und ihre Schnittstellen untereinander sind beschrieben.	☐

Beispiel 1

In dieser Arbeit werden verschiedene Modelle von Checklisten mit integrierten Guidelines beschrieben. Als Grundmodell für eine Checkliste wird das „Kästchen“ - Modell benutzt.

4.2. Modell „Guidelines vor den Checkpunkten“

Dieses Checklistenmodell besteht aus zwei Teilen. In dem ersten Teil sind die Guidelines zu den verschiedenen Checkpunkten vorgestellt. Der zweite Teil umfasst die eigentlichen Checkpunkte, die geprüft werden sollen. Damit man zuordnen kann, welche Guidelines zu welchen Checkpunkten gehören, müssen die Checkpunkte durchnummeriert werden. Zu den verschiedenen Checkpunkten können eine oder mehrere sowie gar keine Guidelines angegeben werden.

Ein Vorteil dieses Modells ist, dass die Inspektoren sich zu der Beantwortung der Checkpunkte vorbereiten können. Vor allem wenn die Gutachter eine Aktivität durchführen sollen, wie z.B. ein Entwurfsdiagramm zu erstellen. Durch ihre Vorbereitung können die Gutachter schnell und sicher die nachfolgenden Checkpunkte prüfen.

In Abbildung 8 ist das Modell „Guidelines vor den Checkpunkten“ vorgestellt und im Anschluss daran finden Sie ein Beispiel zu diesem Modell (Beispiel 2).

Checklist

Guidelines

Zu Checkpunkt 1.
Zu Checkpunkt 2.
.....
.....
.....

Checkpunkte

1. Checkpunkt ☐

2. Checkpunkt ☐

.....
.....
.....

Abbildung 8: Modell "Guidelines vor den Checkpunkten"

Checklist

Guidelines

Zu Checkpunkt 1.
- *Sind Modell, Controller und Viewer auf verschiedene Packages aufgeteilt?*
- *Wurde das Observer-Pattern verwendet?*
- *Ist der Controller eine eigene Klasse, die an das View gekoppelt ist?*

Zu Checkpunkt 2.
- *Ist eine Begründung für die Verwendung angegeben?*

Zu Checkpunkt 3.
- *Entsprechen die Bezechner den Namenskoventionen der Programmiersprache?*
- *Sind alle Bezeichner auf Englisch?*

Zu Checkpunkt 4.
- *Wurde der UML-Standard 2.0 verwendet?*
- *Hat insbesondere das Anwendungsfalldiagramm eine eingezeichnete Systemgrenze?*
- *Sind die Objektnamen immer unterstrichen worden?*

Checkpunkte

1. MVC wurde konsequent umgesetzt. ☐

2. Wurden Design-Patterns richtig verwendet. ☐

3. Sind die Bezeichner konsistent gewählt. ☐

4. UML- Diagramme entsprechen dem UML-Standard. ☐

Beispiel 2

4.3. Modell „Guidelines nach den Checkpunkten“

Dieses Modell sieht dem Modell „Guidelines vor den Checkpunkten“ ähnlich aus – es besteht aus zwei Teilen, wobei hier die Guidelines am Ende der Checkliste positioniert sind.

Checklist

Checkpunkte

1. Checkpunkt ☐

2. Checkpunkt ☐

.....
.....
.....

Guidelines

Zu Checkpunkt 1.
Zu Checkpunkt 2.
.....
.....

Abbildung 9 Modell “Guidelines nach den Checkpunkten“

Checklist

Checkpunkte

1. Die Aufgabe ist beschrieben. ☐

2. Priorisierte Anforderungen des Kunden werden genannt. ☐

3. Use Cases sind vorhanden. ☐

Guidelines

Zu Checkpunkt 1.
• *Achten Sie darauf, dass die Aufgabestellung grob (ca. ein Absatz) beschrieben wurde und die Mission des Projektes ungefähr umrissen ist.*

Zu Checkpunkt 2.
• *Beachten Sie, dass die Anforderungen nach der Wichtigkeit für den Kunden priorisiert sind.*
• *Berücksichtigen Sie, ob es explizit gegeben ist, welche Anforderungen die höchste Priorität und welche die niedrigste Priorität haben.*

Zu Checkpunkt 3.
• *Achten Sie hierbei besonders darauf, dass alle Elemente in den Use Cases beschrieben sind, besonders die wichtigsten Elemente: Akteure, Stakeholder, Systemgrenzen, Erfolgsfall und Misserfolgsfall, Garantie.*
• *Überprüfen Sie, ob es eindeutigen Hauptakteur gibt.*

Beispiel 3

Auch bei diesem Modell, wie auch beim Modell „Guidelines vor den Checkpunkten“, ist ein weiterer Vorteil, dass der aus den Checkpunkten bestehende Teil die kompakte Form der traditionellen Checkliste enthält. Für erfahrene Gutachter, die wenige oder keine Anleitungen brauchen, ist bei diesen zwei Modellen einfach die Guidelines zu überspringen und sich auf die Überprüfung der gegebenen Checkpunkten zu konzentrieren.

Aus der Sicht der Psychologie des Lesens ist das Modell „Guidelines nach den Checkpunkten“ besser als das Modell „Guidelines vor den Checkpunkten“ konstruiert, da beim Lesen eines Dokuments der Blick des Lesers nach unten geht. Deswegen ist für die Inspektoren einfacher und bequemer die zugehörigen Guidelines im Modell „Guidelines nach den Checkpunkten“ zu finden und zu bearbeiten.

4.4. Modell „Guidelines angehängt an die Checkpunkte“

In diesem Modell sind die Guidelines direkt nach jedem einzelnen Checkpunkt gegeben. Bei der Prüfung der Checkpunkte kann der Gutachter diese nacheinander bearbeiten. Ein Nachteil dieses Modell ist, dass die Checkliste aufgebläht wird und ihre kompakte Form verliert.

Abbildung 10 zeigt das Modell “Guidelines angehängt an die Checkpunkte“ und Beispiel 4 ist eine Checkliste zu diesem Modell.

Checklist

Checkpunkt ☐
Guideline
.....
.....
Guideline

Checkpunkt ☐
Guideline
.....
….……..
Guideline
.....
.....
Checkpunkt ☐
Guideline
….……..
….……..
Guideline

Abbildung 10: Modell “Guidelines angehängt an die Checkpunkte“

MVC wurde konsequent umgesetzt. • *Sind Modell, Controller und Viewer auf verschiedene Packages aufgeteilt?* • *Wurde das Observer-Pattern verwendet?* • *Ist der Controller eine eigene Klasse, die an das View gekoppelt ist?*	☐
Wurden Design-Patterns richtig verwendet. • *Ist eine Begründung für die Verwendung angegeben?*	☐
Sind die Bezeichner konsistent gewählt. • *Entsprechen die Bezechner den Namenskoventionen der Programmiersprache?* • *Sind alle Bezeichner auf Englisch?*	☐
UML- Diagramme entsprechen dem UML-Standard. • *Wurde der UML-Standard 2.0 verwendet?* • *Hat insbesondere das Anwendungsfalldiagramm eine eingezeichnete Systemgrenze?* • *Sind Objektnamen immer unterstrichen worden?*	☐

Beispiel 4

4.5. Modell „Online Modell“

Immer mehr werden die Checklisten online gestellt. Deswegen ist hier eine Möglichkeit vorgestellt, wie eine Checkliste mit integrierten Guidelines online aussehen könnte. Das “Online Modell” ist wie das Modell „Guidelines angehängt an die Checkpunkte“ aufgebaut. Zu jedem einzelnen Checkpunkt ist ein Button “Guidelines“ gegeben. Wenn der Gutachter eine Anleitung zu einem bestimmten Checkpunkt braucht, kann er auf dem Button klicken und es wird ein Fenster mit den dazugehörigen Guidelines gezeigt.

Abbildung 11 und Beispiel 5 visualisieren dieses Modell.

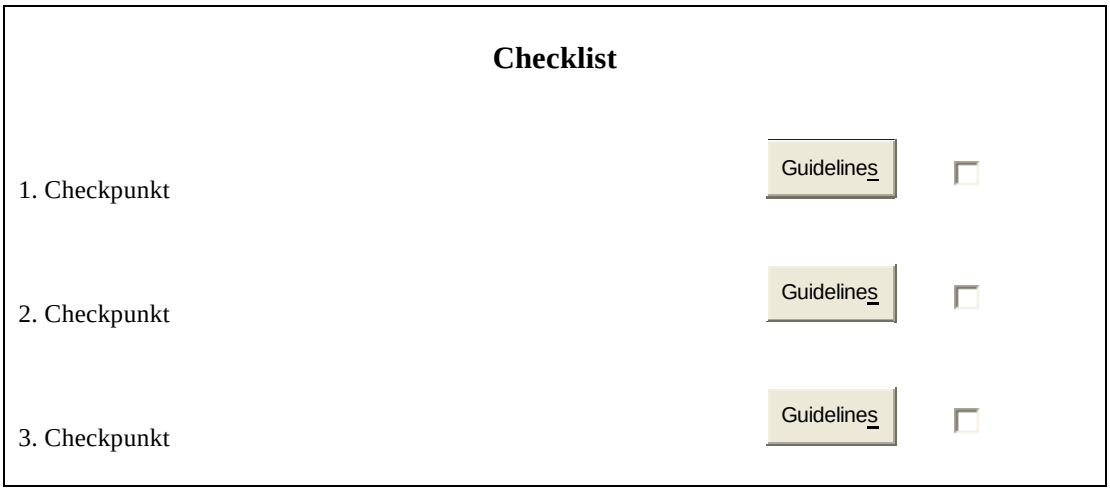
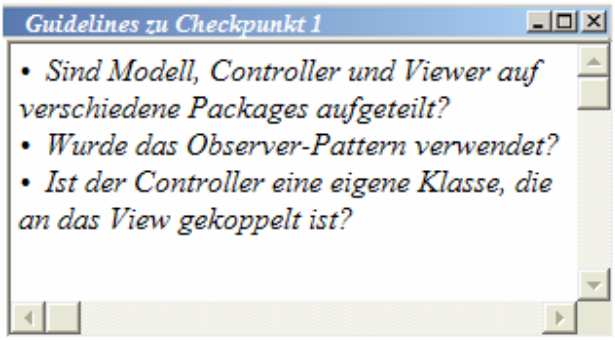


Abbildung 11 “Online Modell”



1. MVC wurde konsequent umgesetzt.	Guidelines	☐
2. Wurden Design-Patterns richtig verwendet.	Guidelines	☐
3. Sind die Bezeichner konsistent gewählt.	Guidelines	☐

Beispiel 5

5. Beschreibung und Auswertung der Umfrage

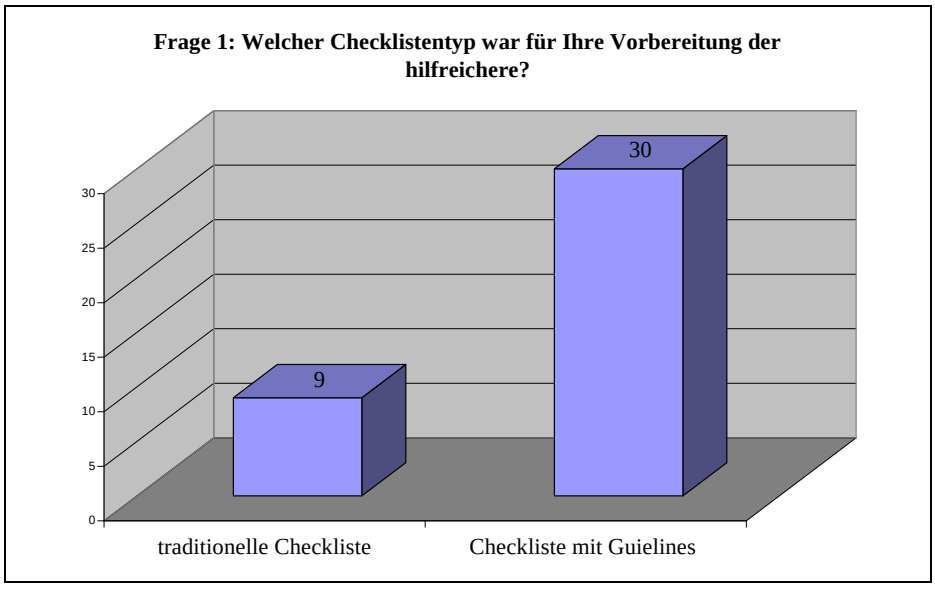
In Rahmen dieser Studienarbeit wurde eine Umfrage durchgeführt. Das Ziel der Befragung war herauszufinden, welcher Checklistentyp (die traditionelle Checkliste oder die Checkliste mit integrierten Guidelines) hilfreicher für die Vorbereitung der Gutachter ist. Andere wichtige Frage war, welches Modell aus den drei in Kapitel 4 beschriebenen Modelle von Checklisten mit integrierten Guidelines ("Guidelines vor den Checkpunkten", "Guidelines nach den Checkpunkten" und "Guidelines angehängt an die einzelne Checkpunkte") am meisten die Inspektoren unterstützt.

5.1. Beantwortungsrate

An der Umfrage nahmen 39 Studenten der Leibniz Universität Hannover teil, 24 von denen waren Informatik Studenten, 9 studieren Mathematik mit Studienrichtung Informatik und 4 waren Studenten von anderen Fächern. Insgesamt 22 von den Befragten Studenten hatten bis jetzt an einem Review als Gutachter teilgenommen, d.h. 56,41 % der Befragten hatten Erfahrung als Gutachter und 43,59 % waren unerfahrene Gutachter. Diejenige Studenten, die bis jetzt an einem Review als Gutachter teilgenommen hatten, haben das an der Leibniz Universität Hannover gemacht.

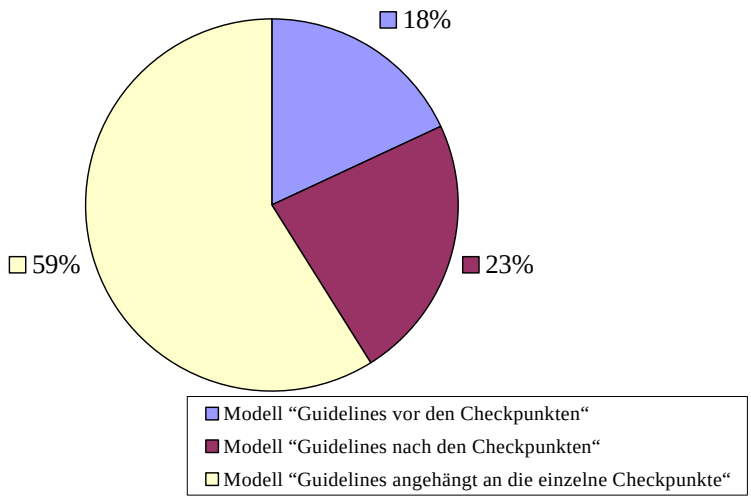
5.2. Auswertung

Wie aus der Abbildung: Frage 1 ersichtlich wird, waren für mehr über die Hälfte der Befragten (77 %) die Checklisten mit integrierten Guidelines hilfreicher für ihre Vorbereitung für die Inspektionssitzung als Gutachter.



Bei der Frage 2 in der Untersuchung (Abbildung: Frage 2) wird nach dem Modell mit Guidelines gefragt, der die Gutachter am meisten unterstützt. Für 59% der Befragten das war das das Modell “Guidelines angehängt an die einzelne Checkpunkte“, für 23 % das Modell “Guidelines nach den Checkpunkten“ und für die restlichen 18 % der Befragten das Modell “Guidelines vor den Checkpunkten“.

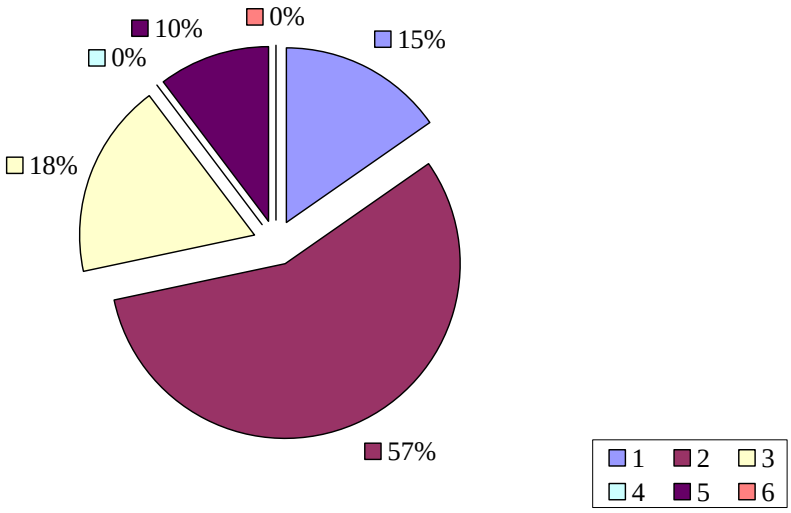
Frage 2: Welches Modell von Checklisten mit Guidelines hat Sie am meisten als Gutachter unterstützt?



Eins der Ziele für die Integration von Guidelines in den Checklisten war ein besseres Verständnis der einzelnen Checkpunkte zu geben, da Checklisten meistens kurz und knapp formuliert sind. Die Abbildung: Frage 3 gibt ein Überblick, wie hilfreich die Guidelines für das Verständnis der einzelnen Checkpunkte für die Untersuchten gewesen sind.

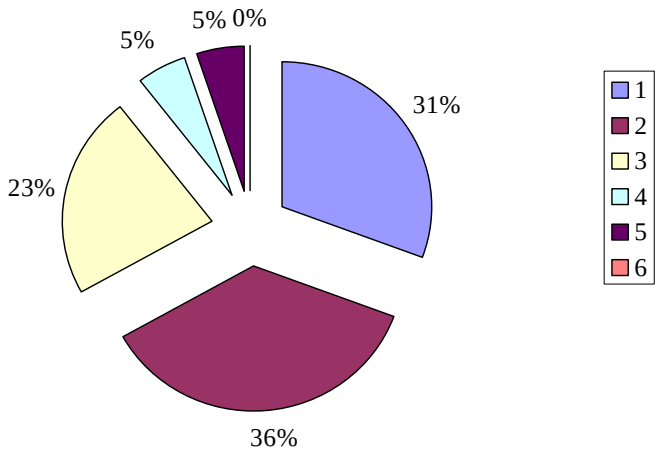
Als Antwortmöglichkeit für diese Frage wurde eine Skala von 1 bis 6 gegeben, wobei 1 für sehr hilfreich ist und 6 für unbedeutsam ist. Keiner der Befragten bestimmt die Integration von Guidelines als unbedeutsam. Für die Mehrheit (57 %) sind die Guidelines hilfreich für das Verständnis der Checkpunkte gewesen und für 15% der Befragte sogar sehr hilfreich.

Frage 3: Wie hilfreich sind die Guidelines für das bessere Verständnis der einzelnen Checkpunkte?

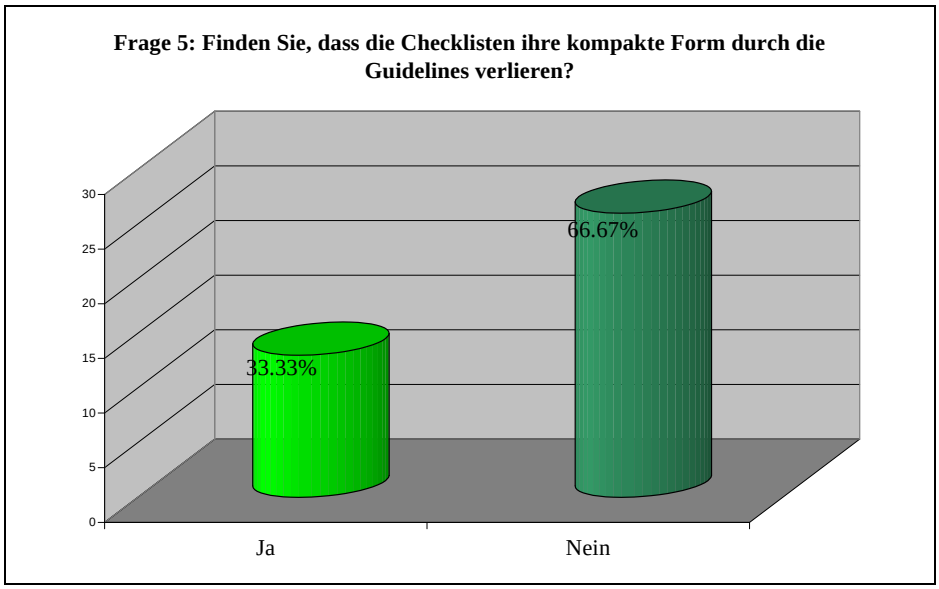


Die nächste Frage ist eine der wichtigsten. Die Teilnehmer der Umfrage wurden gefragt, ob Sie die Integration von Guidelines überhaupt für sinnvoll halten. Als Antwortmöglichkeit für diese Frage wurde ebenso eine Skala gegeben, wobei 1 für sehr sinnvoll ist und 6 für nicht sinnvoll. Wie aus der Abbildung: Frage 4 ersichtlich wird, 31% der Befragten halten die Integration von Guidelines für sehr sinnvoll, 36 % für sinnvoll, 23% für weniger sinnvoll. Nur 10% der Befragten haben sich für die „wenig sinnvolle“ Richtung der Skala entschieden und keiner betrachtet die Guidelines als nicht sinnvoll.

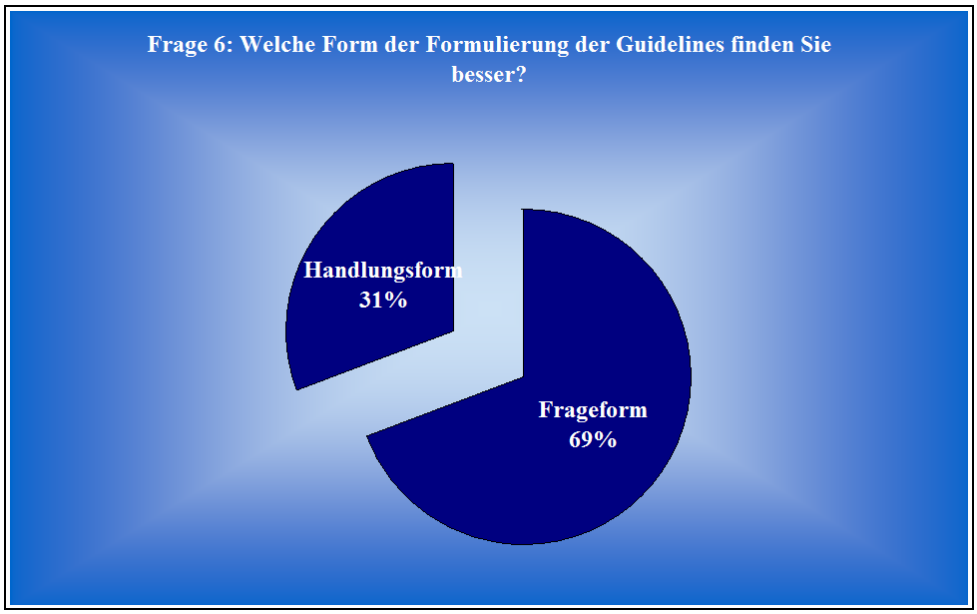
Frage 4: Halten Sie die Integration von Guidelines für sinnvoll?

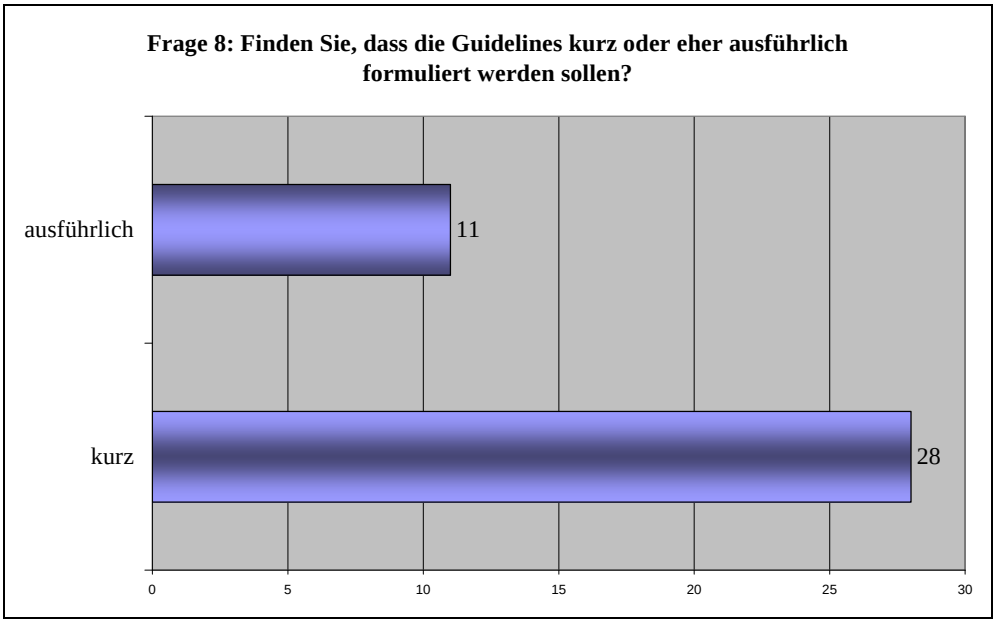


In der Abbildung: Frage 5 sind die Ergebnisse der Frage dargestellt, ob die Befragten finden, dass die Checklisten ihre kompakte Form durch die Guidelines verlieren. Die Mehrheit 26 von 39 Studenten haben mit „Nein“ geantwortet.



Die Fragen 6 und 7 hatten das Ziel herauszufinden, welche Form der Formulierung der Guidelines besser ist und ob sie kurz oder eher ausführlicher formuliert werden sollte. Aus der folgenden Abbildung: Frage 6 wird deutlich, dass 69% der Befragten Studenten die Frageform bevorzugen und Abbildung: Frage 7 zeigt, dass 28 von 39 die kurze Form der Formulierung favorisieren.





Abschließend als Frage 9 wurde eine offene Frage gestellt. Die Studenten wurden nach anderen Vorschlägen, wie Guidelines in Checklisten integriert werden könnten gefragt. Leider wurden keine Einträge gegeben.

6. Zusammenfassung und Ausblick

Die checklistenbasierte Methode stellt die am weitesten verbreitete Lesetechnik dar. Vor allem werden die Checklisten bei den Inspektionen als Lesetechnik hilfreich. Checklisten kommen auch bei anderen Verfahren der Qualitätssicherung zum Tragen, zum Beispiel bei Quality Gates. Ein Nachteil der Checklisten ist, dass sie kurz und knapp formuliert sind und überlassen dem Gutachter zu viel Interpretationsraum. Um dieses Problem zu mildern, wurden im Rahmen dieser Studienarbeit verschiedene Modelle von Checklisten mit Handlungsanleitungen (Guidelines) erarbeitet.

Es wurden drei Modelle von Checklisten mit integrierten Guidelines erarbeitet: “Guidelines vor den Checkpunkten“, “Guidelines nach den Checkpunkten“ und “Guidelines angehängt an die einzelne Checkpunkte“. In der Arbeit wurde eine Umfrage beschrieben, deren Ziel war zu untersuchen, ob die Integration von Guidelines in Checklisten sinnvoll ist und welches der drei Modelle mit Guidelines hilfreicher für die Vorbereitung der Inspektoren ist. Die Ergebnisse haben gezeigt: (a) die Checklisten mit integrierten Guidelines sind hilfreicher als die traditionellen Checklisten ohne Guidelines (b) das Modell „Guidelines angehängt an die einzelne Checkpunkte“ unterstützt über mehr als die Hälfte der Befragten am meisten (c) eine deutliche Mehrheit der Studenten hält die Integration von Guidelines für sinnvoll.

Im Ganzen hat diese kleine Umfrage gezeigt, dass weitere Untersuchungen über die Integration von Guidelines in checklistenbasierten Lesen empfehlenswert sind. Für die Zukunft wäre es also auch interessant zu untersuchen, welche Wirkung die Guidelines auf der Fehlerentdeckungsrate haben, d.h. finden die Gutachter mehr Fehlern, wenn sie Checklisten mit integrierten Guidelines verwenden. Weiterer wichtiger Aspekt zur Untersuchung ist die Auswirkung der Inspektorsfähigkeit, d.h. ob erfahrene und unerfahrene Gutachter unterschiedlich durch Guidelines angeleitet werden müssen. Es lässt sich zusammenfassend feststellen, dass die Erweiterung der Checklisten mit Guidelines eine Verbesserung der checklistenbasierten Lesetechnik verspricht. Diesbezüglich ist die zukünftige Analyse der vorgestellten offenen Fragen als bedeutsam zu betrachten.

Literaturverzeichnis

[1] Basili, V.R., Selby, R.W., “Comparing the effectiveness of software testing techniques.” IEEE Transactions on Software Engineering, 13(12):1278-1296, December 1987.

[2] Basili, V.R., Green, S., Laitenberger, O., Lanubile, F., Shull F., Sorumgard, S. and Zelkowitz, M.V., “The Empirical Investigation of Perspective-Based Reading.” Empirical Software Engineering: An International Journal 1, 2, 1996, pp. 133-164.

[3] Basili, V.R., Caldiera, G., Lanubile, F., and Shull, F., ”Studies on Reading Techniques.” Proc. of the Twenty-First Annual Software Engineering Workshop, SEL-96-002, Greenbelt, MD, December 1996.

[4] Biffl, S., “Software Inspection Techniques to Support Project and Quality Management.“ Austria: Habilitationsschrift, Shaker Verlag, 2001.

[5] Ciolkowski, C., Differding, C., Laitenberger, O., and Muench, J., “Empirical Investigation of Perspective-based Reading: A Replicated Experiment.” International Software Engineering Research Network, Technical Report ISERN-97-13, 1997.

[6] Denger, C., Ciolkowski, M., Lanubile, F.,” Does active Guidance improve Software Inspections? A Preliminary Empirical Study”

[7] Denger, C., Ciolkowski, M., Lanubile, F.” Investigation the Active Guidance Factor in Reading Techniques for Defect Detection”. ISESE 2004.

[8] Ebenau, R.,Strauss, S.,”Software Inspection Prozess”,1993

[9] Fusaro, P., Lanubile, F., and Visaggio, G., “A Replicated Experiment to Assess Requirements Inspection Techniques,” Empirical Software Engineering: An International Journal vol. 2, no. 1, pp. 39-57, 1997.

[10] Fagan, M.E., “Design and code inspections to reduce errors in program development” IBM Systems Journal, vol. 15, no. 3, pp. 219-248, 1976.

[11] Gilb, T., Graham, D., „Software Inspection”, 1993

[12] Halling, M., Biffl, S., Grechenig, T., and Koehle, M., “Using Reading Techniques to Focus Inspection Performance,” Proc. 27th Euromicro Workshop Software Process and Product Improvement, pp. 248- 257, 2001.

[13] Lanubile, F. and Visaggio, G., “Evaluating Defect Detection Techniques for Software Requirements Inspections,” Technical Report no. 00-08, ISERN, 2000.

[14] Launubile, F., Mallardo, F., Calefato, F., Denger, C., and Ciolkowski, M., "Assessing the Impact of Active Guidance for Defect Detection: A Replicated Experiment"

[15] Laitenberger, O., "Experimentelle Bewertung von Software-Lesetechniken in der industriellen Praxis" , April 1996

[16] Laitenberger, O. "Cost-Effective Detection of Software Defects Through Perspective-based Inspections." PhD Theses in Experimental Software Engineering Vol. 1, 2000

[17] Laitenberger, O. and DeBaud, J.-M., "An Encompassing Life Cycle Centric Survey of Software Inspection," J. Systems and Software, vol. 50, no. 1, pp. 5-31, 2000.

[18] Laitenberger, O., Atkinson, C., Schlich, M., and El Emam, K., "An Experimental Comparison of Reading Techniques for Defect Detection in UML Design Documents. ", December 1999

[19] Laitenberger, O., El Emam, K., Harbich, T., "An Internally Replicated Quasi-Experimental Comparison of Checklist and Perspective-based Reading of Code Documents". IEEE Transactions on Software Engineering 27(5): 387–421

[20] Linger, R., Mills, H., and Witt, B.,"Structured Programming: Theory and Practice." Addison-Wesley Publishing Company, 1979.

[20] Miller, J., Wood, M., and Roper, M., "Further Experiences with Scenarios and Checklists," Empirical Software Engineering: An International Journal, vol. 3, no. 3, pp. 37-64, 1998.

[21] Porter, A., Votta, L., and Basili, V.R., "Comparing Detection Methods for Software Requirements Inspection: A Replicated Experiment," IEEE Trans. Software Eng., vol. 21, no. 6, pp. 563-575, June 1995.

[22] Porter, A. and Votta, L., "Comparing Detection Methods for Software Requirements Inspection: A Replication Using Professional Subjects," Empirical Software Eng.: An Int'l J., vol. 3, no. 4, pp. 355-380, 1998

[23] Regnell, B., Runeson, P., and Thelin, T., "Are the Perspectives Really Different?—Further Experimentation on Scenario-Based Reading of Requirements," Empirical Software Engineering: An International Journal, vol. 5, no. 4, pp. 331-356, 2000

[24] Richard C. Linger, Harlan D. Mills, and Bernard I. Witt. "Structured Programming: Theory and Practice." Addison-Wesley Publishing Company, 1979.

[25] Sandahl, K., Blomkvist, O., Karlsson, J., Krysander, C., Lindvall, M., and Ohlsson, N., "An Extended Replication of an Experiment for Assessing Methods for

Software Requirements," Empirical Software Engineering: An International Journal , vol. 3, no. 4, pp. 381-406, 1998.

[26] Schlich, M., "Inspektion des Systemlastenheftes", Eine Publikation des Fraunhofer IESE, Juli 2002

[27] Shull, F., "Developing Techniques for Using Software Documents: A Series of Empirical Studies," PhD thesis, Computer Science Dept., Univ. of Maryland, 1998

[28] Skachkova, M., "Einsatz von Checklisten in der Analytischen Qualitätssicherung", Studienarbeit Juni 2006

[29] Trauboth, H., "Software-Qualitätssicherung" 1996

[30] Thelin, T., Runeson, P., and Wohlin, C., "An Experimental Comparison of Usage-Based and Checklist-Based Reading" IEEE Transactions of Software Engineering, vol.29, no. 8, August 2003

[31] Thelin, T., Runeson, P., and Wohlin, C., "Prioritized Use Cases as a Vehicle for Software Inspections" IEEE Computer Society, 2003

[32] Travassos, G., Shull, F., Fredericks, M., and Basili, V., "Detecting Defects in Object Oriented Designs: Using Reading Techniques to Increase Software Quality" Conference on Object-Oriented Programming, Systems, Languages, and Applications(OOPSLA), Denver, Colorado, 1999.

[33] Winkler, D., Biffl, S., and Thurnher, B., "Investigating the Impact of Active Guidance on Design Inspection"

Anhang

FG Software Engineering
Universität Hannover
Welfengarten 1
3. Stock, Flur G

Fragebogen

SOFTWARE ENGINEERING
UNIVERSITÄT HANNOVER

Stellen Sie sich vor, Sie nehmen an einem Reviewprozess teil. Als Gutachter sollen Sie sich für die Reviewsitzung vorbereiten, indem Sie Einzelgutachten erstellen. Zur Unterstützung der Fehlerentdeckungsphase wird die checklistenbasierte Lesetechnik verwendet. Sie haben eine Checkliste zur Verfügung. Die Checkpunkte geben Ihnen Hinweise darauf, ***was*** *sie in dem Dokument inspizieren sollen und die Guidelines sind Anleitungen,* ***wie*** *Sie die Inspektion durchführen sollten.*
Im Anhang sind zwei Typen von Checklisten vorgestellt. Der erste Typ ist eine traditionelle Checkliste und der zweite Typ ist eine Checkliste mit integrierten Guidelines, von denen es drei verschiedene Modelle gibt.

1. Welcher Checklistentyp war für Ihre Vorbereitung der hilfreichere? *(Bitte nur eins ankreuzen)*

☐ traditionelle Checkliste ☐ Checkliste mit Guidelines

2. Welches Modell von Checklisten mit Guidelines hat Sie am meisten als Gutachter unterstützt? *(Bitte wählen Sie EXAKT EINE der möglichen Antworten/Alternativen aus)*

☐ Modell “Guidelines vor den Checkpunkten“

☐ Modell “Guidelines nach den Checkpunkten“

☐ Modell “Guidelines angehängt an die einzelne Checkpunkte“

3. Wie hilfreich sind die Guidelines für das bessere Verständnis der einzelnen Checkpunkte? *Bitte geben Sie dafür eine Note von 1 bis 6, wobei 1 für sehr hilfreich und 6 für unbedeutsam ist.*

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6

4. Halten Sie die Integration von Guidelines für sinnvoll. *Bitte geben Sie dafür eine Note von 1 bis 6, wobei 1 für sehr sinnvoll und 6 für nicht sinnvoll ist.*

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6

5. Finden Sie, dass die Checklisten ihre kompakte Form durch die Guidelines verlieren?
☐ Ja ☐ Nein
6. Welche Form der Formulierung der Guidelines finden Sie besser? *(Bitte wählen Sie EXAKT EINE der möglichen Antworten/Alternativen aus)*
☐ Die Frageform wie im Modell "Guidelines angehängt an die einzelne Checkpunkte" ☐ Die Handlungsform wie im Modell "Guidelines nach den Checkpunkten"
7. Finden Sie, dass die Guidelines kurz (z.B. nur durch Stichpunkte) oder eher ausführlich formuliert werden sollen? *(Bitte wählen Sie EXAKT EINE der möglichen Antworten/Alternativen aus)*
☐ kurz ☐ ausführlich
8. Haben Sie andere Vorschläge, wie Guidelines in Checklisten integriert werden könnten?
9. Haben Sie bis jetzt an einem Review als Gutachter teilgenommen?
☐ Ja ☐ Nein
Die folgende Frage bitte nur ausfüllen, falls Sie Frage 9 mit JA beantwortet haben.
10. Wo haben Sie an einem Review teilgenommen? (Mehrfachnennungen möglich)
☐ an der Universität Hannover ☐ in einem Privatunternehmen Sonstige:
11. Was studieren Sie?
☐ Informatik ☐ Mathematik mit Studienrichtung Informtik Sonstiges:

Vielen Dank, dass Sie sich Zeit zum Ausfüllen dieses Fragebogens genommen haben!

Beispiel für Szenario aus [15]

Modultest-Szenario
Lesen aus der Sicht eines Modultesters

Sie sollen die Rolle eines Modultesters übernehmen.

Extrahieren Sie dazu den Kontrollflußgraphen aus dem Kodedokument. Fassen Sie dabei geeignete Kodesequenzen (z.B. eine Folge von Anweisungen) zusammen. Achten Sie hierbei besonders darauf, daß alle Verzweigungen im Kontrollflußgraphen vorkommen. Benutzen Sie bei der Erstellung des Kontrollflußgraphen die graphischen Symbole des Formblatts „Symbole für Testszenarien“. Schreiben Sie den Kontrollflußgraphen auf das Formblatt „Ergebnisse Modultest-Szenario“.

Erstellen Sie anhand des Kontrollflußgraphen Testfälle für Entwurf und Kode, mit denen Sie die korrekte Funktionsweise des Moduls intern überprüfen können.

Achten Sie darauf, dass Sie jede Kante des Kontrollflußgraphen und alle Berechnungen im Modul durch entsprechende Testfälle überprüfen! Benutzen Sie dabei nicht unbesehen die Testfälle, die auf der Spezifikation bzw. dem Entwurf vorgegeben sind. Leiten Sie vielmehr die Testfälle von ihrem Kontrollflußgraphen ab. Schreiben Sie die Testfälle auf das Formblatt „Ergebnisse Modultest-Szenario“.

Benutzen Sie die Testfälle als Input für das Modul. Stellen Sie fest, ob das durch den Testfall erzeugte Verhalten (z.B. Ergebnisse von Berechnungen) mit dem in der Spezifikation bzw. Entwurf definierten Verhalten übereinstimmt. Falls Unterschiede vorhanden sind, überprüfen Sie, ob ein Mangel vorliegt oder nicht.

Stellen Sie sich die unten stehenden Fragen während und/oder nach der Ausführung der Aktivitäten und schreiben Sie alle Mängel, die Sie während der Ausführung der Aktivitäten und der Beantwortung der Fragen entdecken, auf das Formblatt „Mängelliste“.

Fragen

Stehen alle Informationen zur Aufstellung eines Testfalls zur Verfügung? (Sind z.B. alle Konstantendefinitionen und Initialisierungen durchgeführt?)

Sind alle Daten im Modul richtig verwendet?

Kann jede Kante des Kontrollflußgraphen durch Testfälle abgedeckt werden?

Werden die Kanten des Kontrollflußgraphen auch korrekt durchlaufen?

Haben Sie alle im Modul auftretenden Formeln durch Testfälle überprüft? Sind die Berechnungen korrekt durchgeführt?