



Fraunhofer Institut
Experimentelles
Software Engineering

Inspektion des Systemlastenheftes



Autoren:
Maud Schlich

Unterstützt durch das BMBF unter
dem Förderkennzeichen VFG0004A
(" QUASAR")

IESE-Report Nr. 036.02/D
Version 1.0
Juli 2002

Eine Publikation des Fraunhofer IESE

Das Fraunhofer IESE ist ein Institut der Fraunhofer-Gesellschaft.

Das Institut überträgt innovative Software-Entwicklungstechniken, -Methoden und -Werkzeuge in die industrielle Praxis. Es hilft Unternehmen, bedarfsgerechte Software-Kompetenzen aufzubauen und eine wettbewerbsfähige Marktposition zu erlangen.

Das Fraunhofer IESE steht unter der Leitung von
Prof. Dr. Dieter Rombach
Sauerwiesen 6
67661 Kaiserslautern

Abstract

In der Forschung ist die Wichtigkeit von Inspektionen als Methode zum frühzeitigen Aufdecken von Problemen und zur Reduzierung von Qualitätskosten unumstritten. Die Dokumentation der Kundenanforderungen mittels Use Cases und anschließende Inspektion der erstellten Dokumente sorgt für eine qualitätsgesicherte Ausgangsbasis für alle nachfolgenden darauf aufbauenden Aktivitäten im Softwareentwicklungsprozess. Daher ist zu diesem frühen Zeitpunkt eine sorgfältige Inspektion besonders wichtig.

In der Literatur existieren nur wenige Anleitungen, wie die eigentliche Fehlersuche optimiert werden kann. So wird die Erstellung von Checklisten bisher nicht, die Erstellung von Szenarien nur eher beiläufig diskutiert. Es gibt bisher auch nur wenig Informationen darüber, wie eine gute Qualität von Checklisten und Szenarien erzeugt werden kann. Allgemein wird der Erfolg in verschiedenen dokumentierten Experimenten auf die jeweils gewählten Lesetechniken zurückgeführt, ohne jedoch die Qualität der Hilfsdokumente zur jeweiligen Lesetechnik verglichen zu haben.

Dieser Bericht konzentriert sich auf die Erstellung von Checklisten und Szenarien für die perspektivenbasierte Lesetechnik. Dazu wird das Agendenkonzept benutzt, das eine systematische Erstellung von Szenarien ermöglicht und die gleichzeitige Zuordnung der zu prüfenden Qualitätsmerkmale zu den jeweiligen Arbeitsaufgaben im Szenario.

Schreibweisen und Begriffe in diesem Dokument

In spitzen Klammern `< >` stehen Parameter, eine Ausnahme bilden die Erweiterungsmechanismen, die in UML mit doppelten spitzen Klammern und hier zusätzlich in *Courier* mit `<<includes>>` und `<<extends>>` dargestellt werden.

Beispiele werden durch **Auszeichnung** hervorgehoben.

Gegenbeispiele werden durch durchgestrichene ~~**Auszeichnung**~~ hervorgehoben.

Inhaltsverzeichnis

1	Einleitung	1
2	Struktur des Systemlastenheftes	2
3	Inspektionen	5
3.1	Der Inspektionsprozess	5
3.2	Lesetechniken und dazu benötigte Dokumente	7
4	Erstellung von Checklisten	12
4.1	Hilfsdokumente für Lesetechniken	12
4.2	Qualität von Checklisten	12
5	Analyse der Qualitätsmerkmale	17
5.1	Allgemeine Qualitätsmerkmale des Systemlastenheftes	17
5.2	Regeln zu den Q-Merkmalen zur Einarbeitung in Dokumentationsrichtlinien	20
6	Checklisten-basierte Inspektion der Use Cases	25
7	Erstellung von Szenarien	27
7.1	Perspektivenbasierte Lesetechnik	27
7.2	Agenden	28
7.3	Agenda zur Erstellung von Szenarien	29
7.4	Beispiel zur Anwendung der Agenda zur Erstellung von Szenarien	30
8	Szenarien für Use Cases in Form von Agenden	35
8.1	Kunde	36
8.2	Systempflichtenheftersteller	38
8.3	Tester	40
9	Zusammenfassung	42
10	Referenzen	43

1 Einleitung

Dieser Bericht behandelt die Vorbereitung von Inspektionen des Anforderungsdokumentes "Systemlastenheft". Welches sich sehr stark auf die Sicht der Kunden konzentriert, daher vorwiegend Use Cases zur Beschreibung der Kundenanforderungen benutzt.

Die Inspektion soll möglichst frühzeitig Probleme für die nachfolgenden Aktivitäten im Softwareentwicklungsprozess vermeiden und ist daher besonders wichtig, um eine solide qualitätsgesicherte Ausgangsbasis zu schaffen. Der eigentliche Inspektionsprozess wird in diesem Dokument unverändert aus bekannten Literaturbeispielen übernommen, die wesentlichen Inhalte dieses Berichtes konzentrieren sich auf qualitativ hochwertige Hilfsdokumente zur Durchführung der Inspektionen. Dazu wurden exemplarisch die checklistenbasierte und die perspektivenbasierte Lesetechnik herangezogen.

In Kapitel 2 wird das Anforderungsdokument "Systemlastenheft" beschrieben und in Kapitel 3 ein Überblick über einen typischen Inspektionsprozess gegeben.

Kapitel 4 erläutert, wie aufbauend auf der Struktur des Systemlastenheftes und dem typischen Inspektionsprozess Hilfsdokumente zur gewählten Lesetechnik erzeugt werden können.

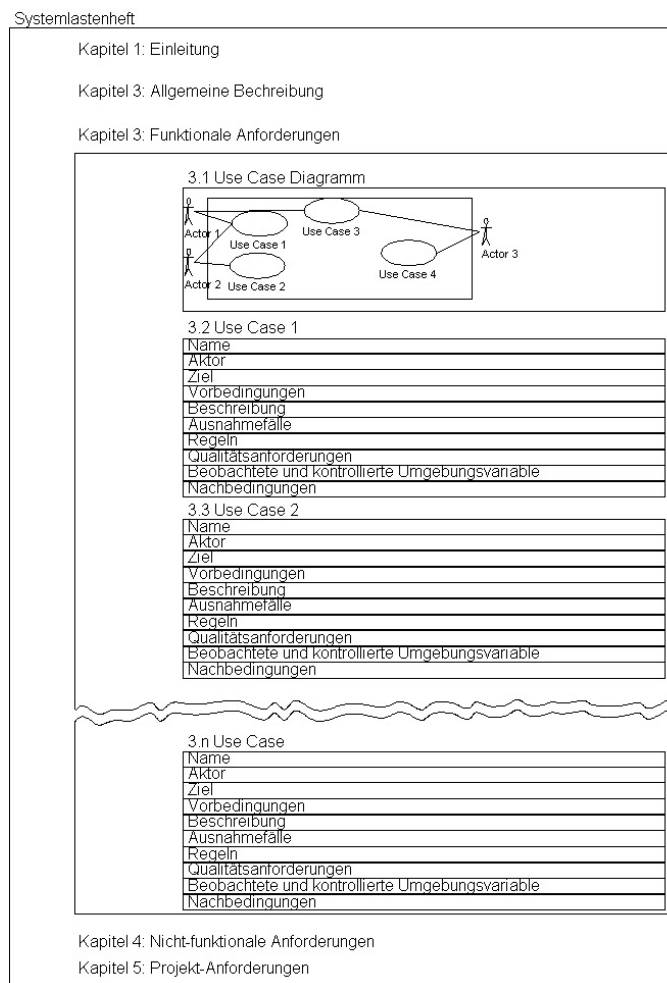
Kapitel 5 – 8 beschreiben detailliert die Vorgehensweise bei der Erstellung der Inspektionsdokumente.

Kapitel 9 fasst die Ergebnisse zusammen.

2 Struktur des Systemlastenheftes

Nach [Kamsties, Knethen, Paech 2001] dient ein Systemlastenheft zur Kommunikation und zur Verhandlung zwischen Kunden, Nutzern und Entwicklern. In der Automobilindustrie werden diese Anforderungen meist von Geschäftsführung oder vorangegangenen Serien ermittelt. Manchmal dienen diese Anforderungskataloge auch zu Verhandlungen mit Unterauftragnehmern.

Die Struktur eines Systemlastenheftes (System Requirements Document) besteht aus mehreren Teilen, wie in Tabelle 2-1: Struktur des Systemlastenheftes zu sehen. Diese Struktur ist in [Kamsties, Knethen, Paech 2001] festgelegt.



Kapitel	Inhalte
1	Einleitung
1.1	Zweck des Dokumentes (wer hat es warum erstellt)
1.2	Geltungsbereich des Systems (Zweck, Kunden, Benutzung, Hauptanwendungsgebiete, hauptsächliche Einschränkungen), ...
1.3	Namenskonventionen (Definitionen, Akronyme, Abkürzungen, ...)
1.4	Referenzen (Standards, Unternehmensweite Dokumentationen, Gesetze, ...)
1.5	Überblick über das gesamte Dokument
2	Allgemeine Beschreibung
2.1	Kontext der Systemnutzung (Serien, Versionen, Releases, Länder, Verkaufs- und Messezeitpunkte, ...)
2.2	Allgemeine Einschränkungen (Produktions- und Verkaufszahlen, sowie typische Einschränkungen der Anwendungsdomäne)
2.3	optional: Charakteristika der Benutzergruppen
3	Funktionale Anforderungen
3.1	Struktur der Funktionalitäten inklusive Begründungen für jeweilige blockweise Zusammenstellung
3.2	Beobachtete und kontrollierte Umgebungsgrößen
3.3	Interaktionsbeschreibungen
4	Nicht-funktionale Anforderungen
4.1	Anforderungen, die sich aus der Produktion ergeben
4.2	Anforderungen, die sich aus dem Anwendungsgebiet ergeben
4.3	Anforderungen, die sich aus dem Einsatzkontext ergeben
4.4	Anforderungen an die Performance (das Antwort-/Zeit-Verhalten, Lastverhalten, Benutzerzahlen, ...)
4.5	Falls im System vorhanden: Anforderungen an die Benutzerschnittstellen (look&feel, Ergonomie (ISO 9241), ...)
5	Projekt-Anforderungen
5.1	Interessengruppen (stakeholder)
5.2	Anforderungen an den Prozess (Techniken, Einschränkungen, ...)
5.3	System- und Nutzungsdokumentation
5.4	Zeitplan (Meilensteine)
5.5	Kostenplan
5.6	Prototyping
5.7	Abnahme- und FreigabeprozEDUREN
5.8	Risiken
5.9	Offene Punkte (, die bis Projektende gelöst werden müssen)
5.10	System- und Softwareteile, die verwendet werden können oder sollen, z.B. Standard-Software (COTS), frühere Versionen

Tabelle 2-1: Struktur des Systemlastenheftes

Dabei sollen Einleitung, allgemeine Beschreibung, nicht-funktionale Anforderungen und Projekt-Anforderungen in natürlicher Sprache verfasst werden, evtl. ergänzt um informelle Diagramme, die den Text illustrieren.

Auch die funktionalen Anforderungen sind in natürlicher Sprache zu beschreiben, wobei es für jeden Block funktionaler Anforderungen (Kapitel 3 des Systemlastenheftes) ein Use Case Diagramm und mehrere dazugehörige Use Case Beschreibungen geben soll.

Zur Darstellung des Use Case Diagramms sollte die UML verwendet werden.

Ein Use Case hat folgende Struktur (in Klammern folgen die Bezeichner in englischer Sprache):

- Name (Name)
- Akteur (Actor)
- Ziel (Intent)
- Vorbedingungen (Precondition)
- Beschreibung (Flow of events)
- Ausnahmefälle (Exceptions)
- Regeln (Rules)
- Qualitätsanforderungen (Quality constraints)
- Beobachtete Umgebungsvariable (Monitored environment variables)
- Kontrollierte Umgebungsvariable (Controlled environment variables)
- Nachbedingung (Post condition)

3 Inspektionen

Eine Inspektion ist eine Methode, um Qualitätseigenschaften von Softwareprodukten zu überprüfen, Fehler in Software Dokumenten zu entdecken und zu korrigieren [Fagan 1976].

In den gesamten Inspektionsprozess geht ein beliebiges Softwareprodukt ein, welches durch u.U. mehrere Zyklen überprüft und korrigiert wird, so dass am Ende des Inspektionsprozesses ein korrigiertes Softwareprodukt herauskommt mit zusätzlichen (zusammengefassten) Informationen über die Inspektion selbst. Die Inspektion ist charakterisiert durch

- einen strukturierten, definierten Prozess
- ein Inspektionsteam aus qualifizierten Personen
- definierte Rollen aller Teilnehmer
- die Verwendung bestimmter Lesetechniken
- dokumentierte Inspektionsergebnisse

3.1 Der Inspektionsprozess

Abbildung 3-1: Der Inspektionsprozess zeigt den typischen Inspektionsprozess, der aus folgenden Phasen besteht:

- Planen (durch die Rolle Organisator dokumentiert in einer Organisationsliste),
- Fehler suchen (durch die Rolle der Inspektoren dokumentiert in Problemlisten mit der Unterstützung einer Lesetechnik),
- Sammeln und Entscheiden (durch die Rollen Moderator, Inspektoren, Autor dokumentiert in einer Fehlerliste bzw. einem Sitzungsprotokoll),
- Korrigieren (durch die Rolle Autor dokumentiert in einer Korrekturliste).

Ziel des Prozesses ist es, aus den Eingangsdokument Softwareprodukt das Ausgangsdokument korrigiertes Softwareprodukt inklusiver der Informationen über den Inspektionsprozess zu erhalten.

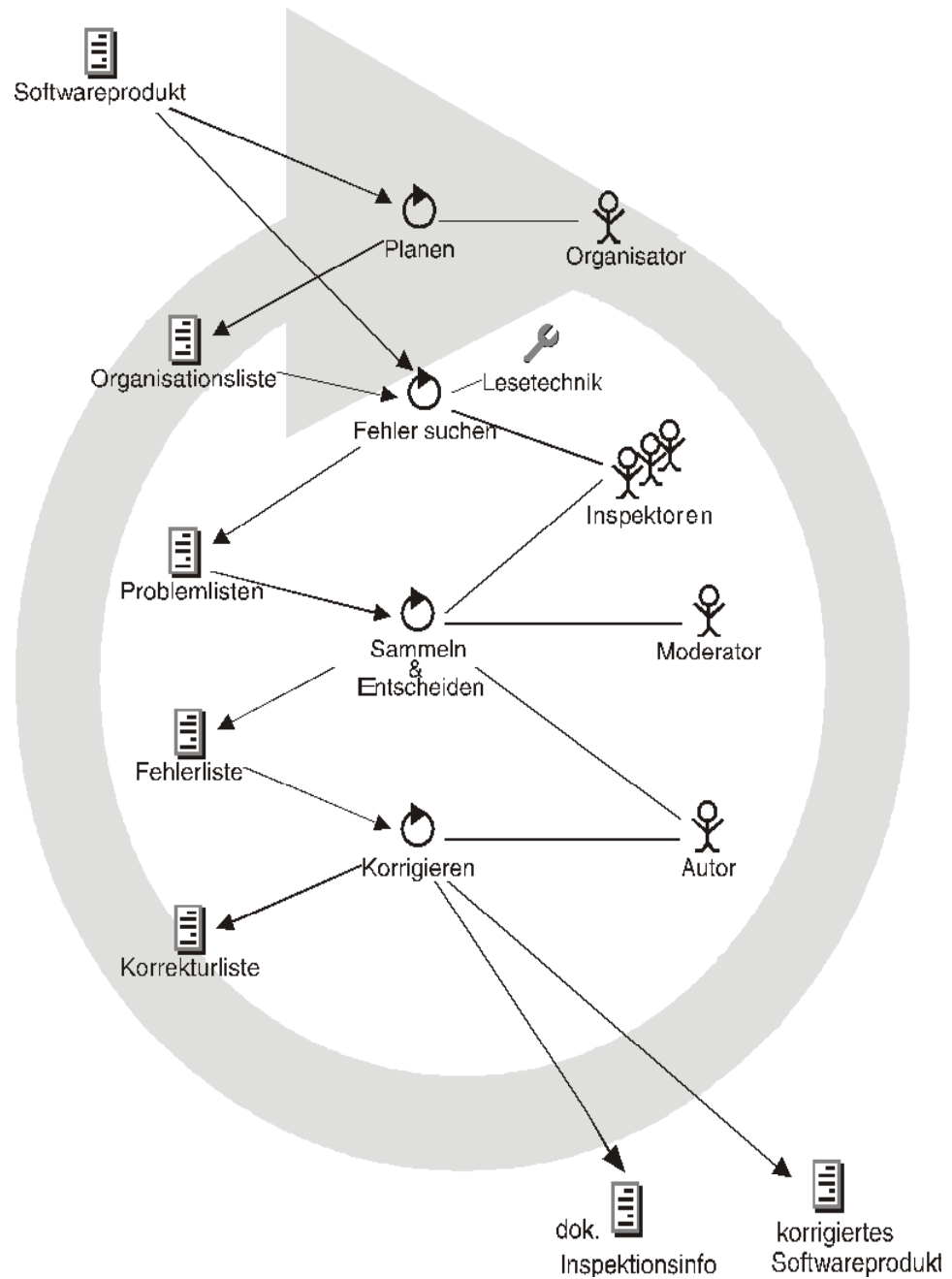


Abbildung 3-1: Der Inspektionsprozess

Inspektionen lassen sich in vier Dimensionen beschreiben [Laitenberger 2000]:



Abbildung 3-2: Die vier Dimensionen des Inspektionsprozesses

3.2 Lesetechniken und dazu benötigte Dokumente

Lesetechniken sind Hilfsmittel für Inspektoren, um Qualitätseigenschaften in Softwareprodukten zu überprüfen. Sie lassen sich nach [Laitenberger 2000] definieren als eine Reihe von Schritten oder Prozeduren deren Zweck es ist, dem Inspektor eine Hilfestellung zu geben, um ein tiefes Verständnis des inspeziierten Software zu erlangen.

Es gibt die nachstehend aufgeführten Lesetechniken:

- Ad-hoc (keine konkrete Vorgehensweise)
- Checklistenbasiertes Lesen
- Reading by stepwise abstraction
- Szenarienbasiertes Lesen
 - Fehlerklassenbasiertes Lesen
 - Perspektivenbasiertes Lesen

Unter Ad-hoc-Lesetechnik lassen sich alle nicht klar definierten Lesetechniken zusammenfassen. Hiermit ist nicht gemeint, dass die Inspektoren das entsprechende Dokument unsystematisch lesen, sondern dass es keine explizite Leseanleitung gibt. Nur selten [Doolan 1992], [Shirey 1992] wird das Ad-hoc-Lesen konkret angesprochen, sehr häufig werden in der Literatur keine Aussagen ü-

ber die angewandte Lesetechnik gemacht, so dass man hier ad-hoc Lesetechnik vermuten kann.

Die Checklistenbasierte Lesetechnik ist zusammen mit der Ad-hoc-Lesetechnik wohl eine der am häufigsten benutzten [Fagan 1976], [Gilb, Graham 1993]. Sie wird in vielen Artikeln anderer Autoren ebenfalls favorisiert z.B. [Ackermann, Buchwald, Lewsky 1989], [Humphrey 1995], [Tervonen 1996]. Die Checklistenbasierte Lesetechnik beruht auf einem Hilfsdokument, welches eine Reihe von Fragen enthält, die der Inspektor während der Problemsuche beantworten muss.

Eine weitere Lesetechnik ist das „Reading by stepwise abstraction“, die nur für abstrahierbare Dokumente anwendbar ist, wozu i.A. Designdokumente [Parnas, Weiss 1985], [Parnas, Weiss 1987] und Codedokumente [Fagan 1976], [Dyer 1992], [Dyer 1992a], [Linger, Mills, Witt 1979] zählen, nicht aber die in diesem Bericht betrachteten Anforderungsdokumente.

Szenarienbasierte Lesetechniken [Basili97] sind neueren Datums. Sie beruhen darauf, dass die Inspektoren eine konkrete Anleitung – ein Szenario – erhalten, um die Problemsuche auf bestimmte Aspekte zu konzentrieren. Dabei wird davon ausgegangen, dass jeder Inspektor ein anderes Szenario bekommt und so das gesamte Inspektionsteam effektiver wird. Die Überlappung der Inspektionsergebnisse der einzelnen Inspektoren soll also geringer, die Überdeckung gleichzeitig größer werden. Dies ist natürlich stark abhängig von der Qualität der Szenarien [Laitenberger 2000].

[Laitenberger 2000]. unterscheidet folgende Charakteristika von Lesetechniken:

- **Anwendungsspektrum**
Auf welche (Arten von) Softwareprodukte(n) ist die Technik anwendbar?
- **Anleitung**
Braucht man zum Anwenden der Technik eine Anleitung?
- **Anpassbarkeit**
Ist die Technik an verschiedene Produkte anpassbar?
- **Wiederholbarkeit**
Lässt sich die Technik ggf. auf das selbe Produkt in der gleichen Weise wiederholt anwenden?
- **Überdeckung**
Deckt die Lesetechnik weitgehend das Auffinden aller Fehler im Produkt ab?
- **Überlappung**
Vermeidet die Technik die Überlappung des Fehlerrückmeldung?

Bei der Auswahl einer Lesetechnik ist es wichtig, sie unter dem Aspekt der Fehlerüberdeckung und –überlappung zu betrachten.

Überdeckung bedeutet dabei, dass die Menge aller Fehler in einem Dokument weitestgehend durch die Inspektion überdeckt werden.

Überlappung bedeutet, dass bei der Arbeit mehrerer Inspektoren an einem Dokument sich deren Bereiche in großem Maße überschneiden, d. h. alle finden gleiche Fehler.

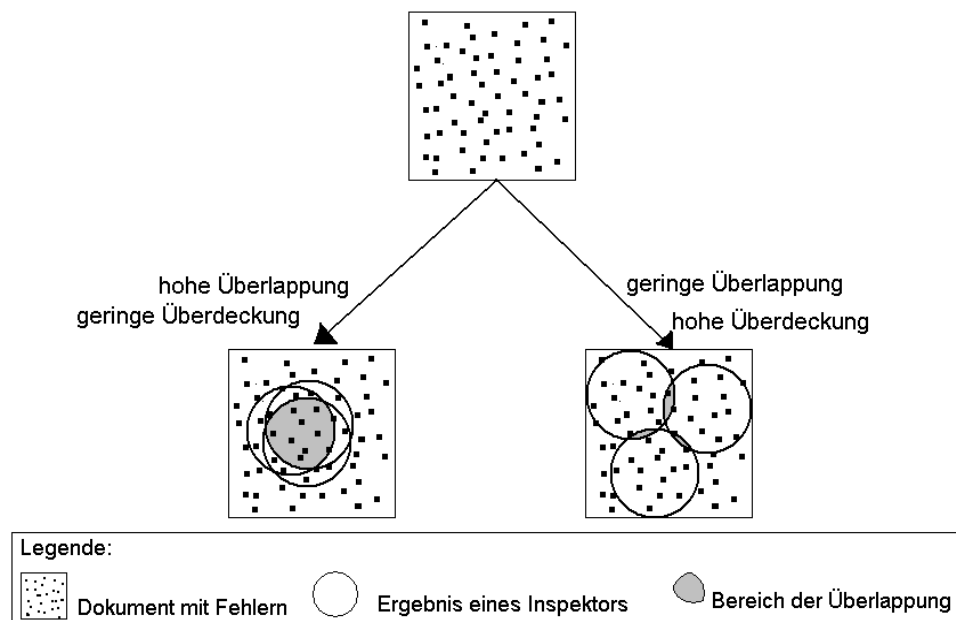


Abbildung 3-3: Überdeckung / Überlappung von Fehlern in der Inspektion

In der untenstehende Tabelle erfolgt eine Bewertung der Lesetechniken nach [Laitenberger 2000] anhand der zuvor aufgeführten Charakteristiken.

Technik	Anwendungs-spektrum	Anleitung	Anpassbar-keit	Wiederhol-barkeit	Über-deckung	Über-lappung
Ad-hoc	Alle Prod.	Nein	Nein	Nein	Fallab-hängig	Fallab-hängig
Checklisten-basiertes Lesen	Alle Prod.	Ja	Ja	Nein	Fallab-hängig	Fallab-hängig
Reading by stepwise abstraction	Alle Prod., die Abstrak-tion erlau-ben	Ja	Nein	Ja	Hoch	Hoch
Fehlerklassen-basiertes Lesen	Alle Prod.	Ja	Ja	Nein	Hoch	Gering
Perspektiven-basiertes Lesen	Alle Prod.	Ja	Ja	Ja	Hoch	Gering

Tabelle 3-1: Charakteristika unterschiedlicher Lesetechniken

Ad-hoc Lesen ist zwar nicht zwangsläufig unsystematisch, hat jedoch keine spezifische Anleitung und ist daher weder anpassbar noch gut wiederholbar. Überlappung und Überdeckung sind ebenso wie beim checklistenbasierten Lesen sehr fallabhängig. Sie beruhen auf den Erfahrungen und Fachkenntnissen der Inspektoren sowie auf deren Fehlerfindungsfähigkeiten. Lesen mit schrittweiser Verfeinerung ist eine sehr intensive Lesetechnik, die daher eine hohe Fehlerauffindungsrate hat und durch die genaue Anleitung auch eine gute Wiederholbarkeit. Allerdings wurde durch Anwendung in Cleanroom Projekten [Linger, Mills, Witt 1979] gezeigt, dass sowohl Überdeckung als auch Überlappung hoch sind. Daher bieten sich Techniken an, die den Inspektoren unterschiedliche an das zu inspizierende Dokument anpassbare Inspektionsaufgaben geben. Hierdurch wird eine geringere Überlappung möglich und eine trotzdem hohe Überdeckung der Fehler im Dokument. Da die Anleitung beim Fehlerklassenbasierten Lesen eher ungenau ist, ist damit auch die Wiederholbarkeit

schlechter als die sehr genauen Aufgaben der perspektivenbasierten Lesetechnik.

Im nachfolgenden Bericht werden die Lesetechniken Checklistenbasiertes Lesen als häufigste systematische Lesetechnik und Perspektivenbasiertes Lesen als unseres Erachtens nach effizienteste Lesetechnik hinsichtlich der Erzeugung der benötigten Hilfsdokumente genauer betrachtet.

4 Erstellung von Checklisten

4.1 Hilfsdokumente für Lesetechniken

Alle Lesetechniken beinhalten eine Prüfung des Softwareproduktes gegen Qualitätsmerkmale. Manchmal werden diese Qualitätsmerkmale in sog. Richtlinien explizit festgehalten. Häufig existieren zumindest Dokumentvorlagen, die ein Minimum an Qualität über die Einhaltung bestimmter Formalien gewährleisten sollen.

Sind Qualitätsmerkmale nicht oder nicht explizit festgehalten, so wird eine Inspektion der Dokumente zwangsläufig nicht wiederholbar sein, da jeder Inspektor auf andere Punkte achten wird. Auch Inspektionsprozesse, die eine Verteilung der Rollen proklamieren, sind letztlich Inspektionen mit ad-hoc Lesetechnik, wenn keine festgeschriebenen Regeln beachtet werden müssen.

Jedes zu inspizierende Softwareprodukt muss allerdings anderen Qualitätsmerkmalen genügen, daher ist der erste Schritt zur Erzeugung von Checklisten wie auch Szenarien der, das zu inspizierende Softwareprodukt genau zu bestimmen. Dies bedeutet, dass ein im Unternehmen allgemeingültiger Name des Softwareprodukt-Typen definiert wird, sowie die Bedingungen für die Erstellung und die Verwendung des Softwareproduktes im Softwareentwicklungsprozess des Unternehmens. Eine wichtige Überlegung, vor allem für die Erstellung von Szenarien, ist die Bestimmung der Nutzer des jeweiligen Dokumentes. Diese Ergebnisse werden typischerweise in einer Richtlinie festgehalten, die aus mehreren einzelnen Regeln und Beispielen zur Erstellung des Dokumenttypes besteht.

4.2 Qualität von Checklisten

In manchen Unternehmen existiert bereits eine solche Richtlinie zur Erstellung des passenden Dokumenttypes, z.B. eine "Richtlinie zur Erstellung von Systemlastenheften". Der Erstellung einer solchen Richtlinie geht in der Regel eine ähnliche Analyse von Qualitätsmerkmalen voraus, wie bei der Erstellung von Checklisten. Manchmal enthalten solche Richtlinien sogar Checklisten, die die Autoren bei der Erstellung der Dokumente unterstützen sollen. Diese Checklisten können selten für die Inspektion übernommen werden, da sie den Kriterien an gute Inspektions-Checklisten meist nicht entsprechen und die typischen Probleme aus der Erfahrung der Inspektoren nicht berücksichtigen.

[Gilb, Graham 1993] fordern, Checklisten so zu erstellen, dass sie direkt auf erstellte Regeln hinweisen, die ebenso wie die Checklisten selbst eine DIN A4-

Seite nicht überschreiten sollen. Dies bedeutet aber folgerichtig, dass diese Checklisten keinen Mehrwert zu den eigentlichen Regeln darstellen, sondern allenfalls besonders häufig gebrochene Regeln hervorheben. Relativiert wird diese Forderung dadurch, dass [Gilb, Graham 1993] empfehlen, den Inspektoren Rollen (z.B. "User", "Tester", "System", "Financial") zuzuweisen und jeder einzelnen Checkliste bis zu vier rollen-spezifischen Fragen hinzuzufügen.

Es gibt allerdings keine allgemein gültige Richtlinie bezüglich der Erstellung solcher Checklisten. Meist beruht sie auf Erfahrungen vorangegangener Inspektionsitzungen oder aus Testergebnissen, also aus vorangegangenen Fehlerfindungsprozessen [Chernak 1996]. Da diese initialen Daten in den meisten Unternehmen nicht vorhanden sind, werden Beispiele aus der Literatur herangezogen, die allerdings selten zu den unternehmensspezifischen Dokumenten passen und letztlich die Erfahrungen anderer Unternehmen darstellen. Solche Checklisten helfen daher nicht bei der Entdeckung von neuen Fehlern, die bisher nicht gefunden wurden, sondern nur beim Auffinden der typischerweise vorkommenden Fehler des jeweiligen Unternehmens. Ein weiteres Problem ist die häufig anzutreffende „Stabilität“ solcher Checklisten, Unternehmen neigen nach unserer Erfahrung dazu, solche Checklisten dauerhaft festzulegen, so dass Sie nicht flexibel an veränderte Prozesse und Fehlerquellen angepasst werden.

Zusätzlich gibt es noch generische Checklisten, die für spezifische Dokumente erstellt wurden [Cockburn 2001] [Quelle von Barbaras Use Case Checklist]. Diese sind allerdings eher zur Erstellung der Dokumente, denn zur Überprüfung geeignet, da sie meist zu umfangreich sind, z.B. enthält die Checkliste [Cockburn 2001] 31 Fragen. Viele Fragen verhindern die Konzentration auf die eigentlich wichtigen Probleme des Dokumentes und führen so häufig zu einer Entdeckung vieler unwichtiger Fehler [Parnas, Weiss 1987]. In der Untersuchung von mehr als 100 Checklisten durch [Bryczynski 1999] behandeln die Checklisten zu Anforderungsdokumenten typischerweise Fragen zu Konsistenz, Korrektheit und Vollständigkeit der Anforderungen. Dabei ist es problematisch, wenn die gestellten Fragen zu allgemein sind: "Are software requirements clear and ambiguous?" [SPAWAR 1997 cit. Bryczynski 1999].

[Gilb Graham 1993] haben in ihrem Buch rollenspezifische Checklisten ermittelt, um dem Problem entgegen zu wirken: dass jeder Inspektor das gesamte Dokument mit den gleichen Fragen liest und somit theoretisch auch jeder Inspektor die gleichen Fehler entdeckt. Diese Charakteristik von Lesetechnik wird Überlappung genannt (s. a. Abbildung 3-3: Überdeckung / Überlappung von Fehlern in der Inspektion). Sie ist bei der checklistenbasierten Lesetechnik abhängig von der tatsächlichen Technik des Lesenden, da trotz einer einheitlichen Checkliste jeder Inspektor das Lesen des Dokumentes und die Beantwortung der Fragen unterschiedlich durchführen kann. Im Extremfall kann der Lesende das gesamte Dokument ad-hoc lesen und anschließend die Fragen der Checkliste beantworten, ohne weitere Fehler aufzudecken. Der Nachweis der durchgeführten Fehlersuche besteht bei der checklistenbasierten Lesetechnik im

Abhaken der Fragen, wofür nach der Analyse von [Bryczynski 1999] lediglich bei 2 von 117 untersuchten Checklisten eine Möglichkeit vorgesehen wurde. Es wird kein weiteres Arbeitsergebnis außer der Liste von gefundenen Problemen gefordert. Die Lesetechnik ist daher nicht kontrollierbar, die Wiederholbarkeit der Ergebnisse hängt sehr stark vom Inspektor selber ab.

Ein weiteres Problem stellt die mangelnde Arbeitsanleitung für die Inspektoren dar. Die Aufgabe der Inspektoren lautet meist:

- 1.) Lies die erste Frage der Checkliste
- 2.) Lies das gesamte Dokument und beantworte die Frage
- 3.) Lies die nächste Frage
- 4.) Gehe zu 2.)

Oder alternativ:

- 1.) Lies das gesamte Dokument
- 2.) Beantworte alle Fragen der Checkliste.

In jedem Fall wird häufig davon ausgegangen, dass alle Inspektoren das gesamte Dokument lesen müssen. Dies führt meist zu Zeitproblemen, da die Dokumente häufig nicht in der für die Inspektion zur Verfügung stehenden Zeit gelesen werden können. [Graham 2000] empfiehlt daher, statt oberflächlichem Lesen des gesamten Dokumentes nur wenige Seiten sehr konzentriert zu lesen, um die "tief liegenden Fehler" aufzudecken. Sie macht allerdings keine Aussage darüber, wie die wenigen Seiten ausgewählt werden können.

Checklisten haben eine hohe Qualität, wenn sie

- genau zu dem zu inspizierenden Dokument passen
- die Arbeit der Inspektoren (und nicht der Autoren) unterstützen
- die geforderten Qualitätsmerkmale überprüfen helfen
- zu einer hohen Fehlerentdeckung beitragen

Dazu müssen Sie z.B. folgende äußere Eigenschaften aufweisen (s. a. [Bryczynski 1999]):

- Umfang nicht länger als eine DIN A4-Seite
- genügend Raum zum Notieren von Bemerkungen

- Raum zum Abhaken der analysierten Fragen
- Übersichtlichkeit durch Strukturierung der Fragen in Blöcken

Die nachfolgende Abbildung 4-1: Agenda zur Erstellung von Checklisten beschreibt eine mögliche Vorgehensweise zur Erstellung von Checklisten, die diese Qualität erzeugen soll. Zusammenfassend lässt sich sagen, dass eine gute Checkliste aus der Erfahrung des Unternehmens, wie sie z.B. in Dokumentationsrichtlinien, Messergebnissen und Fehleranalysen sowie aufgelisteten Qualitätsmerkmalen enthalten sind, extrahiert wird und daher einer häufigen Überarbeitung bedarf.

Das Kapitel 5 erzeugt eine solche Checkliste gemäß der beschriebenen Vorgehensweise. Dazu wird nach der Bestimmung des Dokumenttypes "Systemlastenheft" - falls vorhandenen - die zugehörige Dokumentationsrichtlinie analysiert. Aufbauend aus vorhandenen Erfahrungen und Messwerten sowie den Ergebnissen der Analyse werden die Qualitätsmerkmale des Dokumentes aufgelistet. Die Qualitätsmerkmale werden in spezifischen Regeln konkretisiert, die entsprechend in der Dokumentationsrichtlinie Eingang finden sollten.

Das blockweise Zusammenfassen einzelner Qualitätsmerkmale ermöglicht eine Konzentration auf die wesentlichen Eigenschaften des zu inspizierenden Dokumentes und vereinfacht den nachfolgenden Schritt der Erstellung passender Checklisten-Fragen.

No	Step	Validation								
1	Bestimme den Typ des Software-Dokumentes und benenne ihn eindeutig. Name: <Dokumenttyp>	Es gibt im Unternehmen kein anderes Dokument mit dem gleichen oder einem sehr ähnlichen Namen; es gibt für diesen Dokumenttyp nur genau den einen festgelegten Namen.								
2	Analysiere - falls vorhanden - die Erstellungsrichtlinie zu <Dokumenttyp> und liste die typischen Qualitätsmerkmale auf.	Zu jedem ermittelten Qualitätsmerkmal ist ein eindeutiger Verweis auf die Erstellungsrichtlinie zu <Dokumenttyp> vorhanden.								
3	Analysiere typische Fehler aus vorhandenen Messdaten und aus Erfahrung und vervollständige dahingehend die Liste der Qualitätsmerkmale.	Zu jedem typischen Fehler ist der Verweis auf ein Beispiel vorhanden.								
4	Fasse thematisch ähnliche Qualitätsmerkmale blockweise zusammen und benenne diesen Block. Sortiere die entstandenen Blöcke nach Priorität aufgrund der Liste von typischen Fehlern (der wichtigste zuerst). Streiche alle Blöcke, deren Inhalte durch automatisierte Tools und nicht durch Inspektoren geprüft werden.	Der Name des Blocks gibt Hinweise darauf, auf welchen Teil von <Dokumenttyp> sich die Fragen beziehen oder auf welche Art von Qualitätsmerkmalen.								
5	Erarbeite zu den ersten drei bis vier Blöcken von Qualitätsmerkmalen ein bis drei geschlossene Fragen und trage Sie in die Checkliste ein. Alle anderen Blöcke werden nicht weiter betrachtet. <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 100px; height: 20px;"></td> <td style="width: 20px; text-align: center;">➔➔</td> </tr> <tr> <td style="text-align: center;"><Block1></td> <td style="width: 20px;"></td> </tr> <tr> <td style="text-align: center;"><Frage1> <Platz für Bemerkung></td> <td style="width: 20px;"></td> </tr> <tr> <td style="text-align: center;"><Frage2> <Platz für Bemerkung></td> <td style="width: 20px;"></td> </tr> </table>		➔➔	<Block1>		<Frage1> <Platz für Bemerkung>		<Frage2> <Platz für Bemerkung>		<p>Jede Frage lässt sich eindeutig mit Ja oder Nein beantworten.</p> <p>Jede Frage unterstützt die Suche nach typischen Fehlern zu den gelisteten Qualitätsmerkmalen.</p> <p>Die Checkliste kann auf einer DIN A4 Seite gedruckt werden (min. 10 pt Schrift) und enthält Platz zum Abhaken der Fragen und zum Notieren von Bemerkungen je Frage.</p>
	➔➔									
<Block1>										
<Frage1> <Platz für Bemerkung>										
<Frage2> <Platz für Bemerkung>										

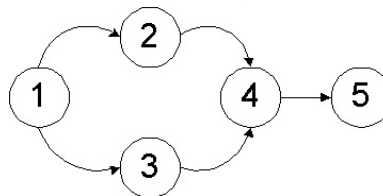


Abbildung 4-1: Agenda zur Erstellung von Checklisten

5 Analyse der Qualitätsmerkmale

5.1 Allgemeine Qualitätsmerkmale des Systemlastenheftes

Gute Anforderungsdokumente erfüllen folgende allgemeine Qualitätsmerkmale [IEEE 830]:

- Korrektheit
- Eindeutigkeit
- Vollständigkeit
- Konsistenz
- Charakterisierung der Wichtigkeit / Dauerhaftigkeit
- Prüfbarkeit
- Änderbarkeit
- Verfolgbarkeit

Tabelle 5-1: Allgemeine Qualitätsmerkmale von Anforderungsdokumenten analysiert diese Qualitätsmerkmale näher (Allgemeine Bedeutung) und legt fest, wie diese Qualitätsmerkmale für das Systemlastenheft umgesetzt werden, bzw. welche Einschränkungen es für das Systemlastenheft gibt (Besonderheit für Systemlastenheft). Aus dieser Analyse wird dann die Richtlinie zur Erstellung von Use Cases abgeleitet, wobei die Besonderheiten für das Systemlastenheft explizit berücksichtigt werden.

Qualitätsmerkmal	Allgemeine Bedeutung	Besonderheit für Systemlastenheft
Korrektheit	Jede enthaltene Anforderung repräsentiert etwas, was das System tatsächlich leisten soll bzw. was einem Benutzerbedürfnis entspricht.	Korrektheit im angegebenen Sinn, lässt sich nur durch die aktive Beteiligung der Systemnutzer und der Systemkunden (Stakeholder) erreichen. Dazu sollten diese mindestens bei der Erstellung und möglichst auch bei der Inspektion mitwirken ¹ .

¹ Daher zählen zu den potenziellen Nutzern des Dokumentes alle Stakeholder. Diese werden z.B. beim perspektivenbasiertem Lesen durch eine Rolle und dazugehörigem Szenario repräsentiert (s. Agenda zur Erstellung von Szenarien)

Qualitätsmerkmal	Allgemeine Bedeutung	Besonderheit für Systemlastenheft
Eindeutigkeit	<p>(Lesbarkeit, Verständlichkeit)</p> <p>Eine allgemein bekannte Sprache und Notation werden verwendet.</p> <p>Die Übersicht über das Dokument bleibt gewahrt. Für denselben Sachverhalt wird an verschiedenen Stellen immer derselbe Begriff verwendet. Alle nicht-trivialen Begriffe, insbesondere bei Gefahr der Mehrdeutigkeit, sind im Glossar erläutert.</p> <p>Verschiedene Interpretationsmöglichkeiten zum selben Sachverhalt an verschiedenen Stellen sind nicht denkbar.</p>	<p>Ein explizites Glossars wird in der Struktur des Systemlastenheftes nicht gefordert. Begriffe werden unter Punkt 1.3 Namenskonventionen (s. [Kamsties, Knethen, Paech 2001]) erklärt.</p>
Vollständigkeit	<p>Alles, was das System machen soll und alle Reaktionsweisen auf die möglichen Klassen von Eingabe- und Ausgabedaten (gültige und ungültige) ist mit dem gewünschten Detaillierungsgrad spezifiziert.</p>	<p>Die Vollständigkeit der Eingabe- und Ausgabedaten wird im Beispiel-Systemlastenheft durch die vollständige Auflistung der "Monitored Environmental Variables" und der "Controlled Environmental Variables" erfüllt. Die Vollständigkeit basiert dabei aus Sicht der Stakeholder nur auf die Variablen, die zum S-Level gehören² (s. [Kamsties, Knethen, Paech 2001]).</p>
Konsistenz	<p>Es existieren keine Anforderungen die miteinander in Konflikt stehen / sich widersprechen.</p>	<p>Insbesondere gibt es keine Use Cases die unterschiedliches Verhalten des Systems fordern.</p>
Charakterisierung der Wichtigkeit	<p>Alle Anforderungen sind bezüglich ihrer Wichtigkeit für den Nutzer / den Kunden gekennzeichnet.</p>	<p>Eine solche Charakterisierung ist im Systemlastenheft derzeit nicht vorgesehen, wird aber als Attribut in der Änderungsdatenbank gefordert werden. [von Knethen]</p>
Charakterisierung der Dauerhaftigkeit	<p>Alle Anforderungen sind bezüglich ihrer Dauerhaftigkeit (Anzahl erwarteter Änderungen, Fristen Wahrscheinlichkeiten) gekennzeichnet.</p>	<p>Eine solche Charakterisierung ist im Systemlastenheft derzeit nicht vorgesehen, wird in [von Knethen] behandelt.</p>

² Die eigentliche Vollständigkeit wird erst im Systempflichtenheft aus Systemsicht vorhanden sein.

Qualitätsmerkmal	Allgemeine Bedeutung	Besonderheit für Systemlastenheft
Prüfbarkeit	<p>Es gibt eine realistische Methode, um zu überprüfen, ob das zu erzeugende System die Anforderung erfüllt, d.h. die Anforderungen sind messbar.</p> <p>Die enthaltenen Anforderungen sind frei von Subjektivität.</p>	Die Validierbarkeit lässt sich besonders gut bei der Inspektion - insbesondere bei der perspektivenbasierten Lesetechnik durch die Rolle des Testers – feststellen, da hierbei überprüft wird, ob zu jeder Anforderung auch ein Testfall erstellt werden kann.
Änderbarkeit	<p>Struktur und Stil ermöglichen eine Änderung, ohne dass Vollständigkeit oder Konsistenz verletzt werden.</p> <p>Anforderungen werden möglichst nur an einer Stelle beschrieben, um Redundanzen zu vermeiden. Sofern zusammenhängende Anforderungsteile an verschiedenen Stellen im Dokument beschrieben werden, so müssen diese Stellen einen entsprechenden Querverweis auf alle anderen Stellen enthalten.</p>	Wird in [von Knethen] behandelt.
Verfolgbarkeit	Anforderungen sind verfolgbar zu früheren Entwicklungsphasen (Rückwärtsverfolgbarkeit) und zu nachfolgenden Entwicklungsphasen (Vorwärtsverfolgbarkeit)..	<p>Wird in [von Knethen] ausführlich behandelt.</p> <p>Eine Rückwärtsverfolgbarkeit ist beim Systemlastenheft nur bedingt zu evtl. Vorgänger-Dokumenten (z.B. Produktskizzen, Markt- und Vorstudien) möglich, wird aber nicht explizit vorgegeben.</p>

Tabelle 5-1: Allgemeine Qualitätsmerkmale von Anforderungsdokumenten

5.2 Regeln zu den Q-Merkmalen zur Einarbeitung in Dokumentationsrichtlinien

Nachfolgend werden einige typische Regeln aus den Qualitätsmerkmalen abgeleitet, die für alle Teile des Systemlastenheftes relevant sind. Diese Regeln werden zusammengefasst und ergeben mit Beispielen versehen eine Richtlinie zur Erstellung von Systemlastenheften / Kapitel Use Cases oder münden in einer Überarbeitung eines bereits vorhandenen Dokumentes. In Quasar wurde initial ein solches Dokument namens " erstellt, um die prinzipielle Struktur aller Anforderungsdokumente festzulegen. Anschließend wurde das Dokument nach den vorliegenden Qualitätsmerkmalen überarbeitet.

Q-Merkmal	Regeln
Eindeutigkeit	Alle Teile des Systemlastenheftes sind mit Text gefüllt oder tragen die Kennzeichnung "entfällt".
Vollständigkeit	Ein Use Case ist ausführlich genug beschrieben, wenn er alle möglichen Instanzen des Use Cases abdeckt, d.h. alle denkbaren Szenarien müssen im Use Case subsummiert sein.
Eindeutigkeit: Lesbarkeit	Alle möglichen Instanzen von Use Cases, die das gleiche Ziel verfolgen werden in einem Use Case beschrieben. Falls dadurch der Use Case zu komplex / zu lang (> 2 DIN A4 Seiten) würde, kann er durch <<includes>> Beziehungen aufgespalten werden. ³

³ Use Cases lassen sich besonders sinnvoll an den Stellen aufspalten, an denen eine zeitliche Pause zwischen zwei Aktionen liegt.

Use Case Feld	Q-Merkmal	Regeln
Name	Eindeutigkeit	Der Name muss der Grammatik: <Objekt ⁴ > <Prädikat> genügen ⁵ .
	Korrektheit / Vollständigkeit	Die Wahl der <Objekte> und <Prädikate> sollte immer die Aktivität des initiierenden Aktors widerspiegeln ⁶ .
	Vollständigkeit / Konsistenz	Hinter dem Namen sollte in Klammern ein evtl. Erweiterungsmechanismus in der Form (<Erweiterung> <anderer Use Case Name>) angegeben werden ⁷ .
Aktor	Korrektheit	Alle Aktoren müssen eine Rolle oder ein externes System ⁸ repräsentieren. ⁹
	Korrektheit	Rollen müssen geschlechtsneutral ¹⁰ und personenneutral ¹¹ bezeichnet werden.
	Korrektheit	Externe Systeme sollten herstellernerneutral ¹² und technologieneutral ¹³ bezeichnet werden.
Ziel	Korrektheit	Das Ziel beschreibt das, was aus Sicht des initiierenden Aktors erreicht werden soll. ¹⁴
	Eindeutigkeit	Das Ziel ist in der Vergangenheitsform beschrieben, so als wäre die Beschreibung nachvollzogen worden und das Ziel erreicht.
	Validierbarkeit	Das Ziel muss realistisch und eindeutig messbar sein (ja/nein). ¹⁵

⁴ mit <Objekt> ist das grammatikalische Objekt im Sinne von <Subjekt> <Prädikat> <Objekt> gemeint, nicht das Objekt im Sinne von <Instanz einer Klasse> im Kontext von Software-Engineering bzw. Objektorientierung. Trotzdem kann natürlich das <Objekt> auch in der nachfolgenden Objektanalyse zu einer <Klasse> oder der <Instanz einer Klasse> werden.

⁵ z.B. **Sitzplatz reservieren, Benutzer anmelden**. Falsch wäre: **Sitzplatzreservierung** (Substantiierung von Prädikaten) oder **Anmelden** (Auslassung des Objektes)

⁶ Checkliste für Use Cases von Dr. rer. nat. habil Barbara Paech, Absatz **Name**, Regel 1.

⁷ z.B. **Sitzplatz reservieren (<<includes>> freien Sitzplatz suchen)**

⁸ Externe Systeme sind z.B. auch beobachtete Variablen (monitored variables), die typischerweise ein Verhalten auslösen.

⁹ Checkliste für Use Cases von Dr. rer. nat. habil Barbara Paech, Absatz **Aktor**, Regel 1.

¹⁰ z.B. **Personalleitung, Projektmanagement, Entwickler/-in**

¹¹ nicht: **Frau Müller** oder **Otto**

¹² nicht **Microsoft Word**, sondern **Textverarbeitung**

¹³ nicht **relationale Datenbank** oder **objektorientierte Datenbank**, sondern **Datenbank**

¹⁴ Checkliste für Use Cases von Dr. rer. nat. habil Barbara Paech, Absatz **Ziel**, Regel 1.

¹⁵ z.B. **Ein Sitzplatz wurde reserviert**.

Vorbedingungen	Korrektheit	Die Vorbedingung beschreibt, was aus Sicht des initiiierenden Ak-tors nötig ist, um das Ziel zu erreichen und die Schritte der Be-schreibung durchführen zu können.
	Korrektheit	Die Vorbedingung muss innerhalb der Systemgrenzen liegen.
	Eindeutigkeit	Die Vorbedingung ist in der Gegenwartsform beschrieben.
	Vollständigkeit / Validierbarkeit	Die Vorbedingung enthält die genaue Beschreibung des Zustandes des <Objektes> und des gesamten Systems.
Beschreibung	Korrektheit / Vollständigkeit / Eindeutigkeit / Validierbarkeit	Die Beschreibung enthält Sätze, die abwechselnd die Aktion des Aktors und daraufhin die Reaktion des Systems beschreiben. Der erste Satz ist immer eine Aktion des Aktors. Nach der letzten Antwort des Systems erfolgt maximal ein weiterer Satz, der die Aktion des Aktors außerhalb der Systemgrenzen be-schreibt. ¹⁶ Die Beschreibung enthält die Änderung einer oder meh-erer kontrollierter Größen.
	Eindeutigkeit	Die Sätze der Beschreibung sind innerhalb eines Use Cases fortlau-fend nummeriert.
	Eindeutigkeit	Die Sätze enthalten maximal einen Nebensatz.
	Eindeutigkeit	Die Sätze genügen den definierten Standardsätzen ^{17 18} .
	Eindeutigkeit	Jeder Satz der Beschreibung ist in der Gegenwartsform beschrieben mit Ausnahme von Verweisen auf zurückliegende Ereignisse, die in der Vergangenheitsform formuliert werden.
	Konsistenz	Ausnahmefälle werden in die Beschreibung mit ihrem Namen nur als Verweis auf den Use Case Teil Ausnahmefälle aufgenommen.

¹⁶ Das System hat immer das letzte Wort: Keine Aktion des Aktors bleibt ohne Systemantwort, wenn sie in-nerhalb der Systemgrenzen passiert.

¹⁷ s. a. [Rupp 2001], [Kamsties, Berry, Paech 2001]

¹⁸ Hierzu sollten eine Reihe von Standardverben im Glossar definiert werden: z.b. **pflegen, verwalten, ein-geben, aufrufen** und eine entsprechende Grammatik, sowie Schlüsselwörter festgelegt werden, die dann zu Standardsätzen gebildet werden:

<Subjekt> <Prädikat> <Objekt>.

Falls <Objekt> <erfüllte Bedingung>, **dann** <Subjekt> <Prädikat> <Objekt>.

Für alle <Objekte> in <Objektmenge>: <Subjekt> <Prädikat> <Objekt>

s.a. "Listen deutscher und englischer Prozesswörter" in [Rupp 2001]

Nachbedingung	Korrektheit	Die Nachbedingung beschreibt das erreichte Ziel innerhalb der Systemgrenzen und evtl. Folgerungen für den Akteur, die außerhalb der Systemgrenzen liegen. ¹⁹
	Vollständigkeit	Hierzu zählen auch alle Endzustände der <Objekte>, diese müssen genau die gleichen Zustandsmerkmale betreffen wie in der Vorbedingung.
	Eindeutigkeit / Validierbarkeit	Die Nachbedingung ist in der Vergangenheitsform beschrieben, so als wäre die Beschreibung nachvollzogen worden und das Ergebnis läge bereits vor.
Ausnahmefälle	Eindeutigkeit	Jeder Ausnahmefall genügt der Grammatik: <Name>: <Standardhalbsatz für Bedingung> System <Prädikat> <Objekt>. <evtl. weitere Standardsätze>.
	Eindeutigkeit / Validierbarkeit	Jeder Name eines Ausnahmefalles entspricht einer kurzen Aussage, die wahr sein muss, damit die Ausnahme gültig ist. ²⁰
	Eindeutigkeit / Konsistenz	Jeder Name eines Ausnahmefalles ist innerhalb des Use Cases einmalig. ²¹
	Eindeutigkeit	Jeder Ausnahmefall ist in der Gegenwartsform beschrieben.
Regeln	Eindeutigkeit	Jede Regel genügt der Grammatik: <Zeitpunkt> ²² : <Subjekt> <Prädikat> <Objekt>.
	Eindeutigkeit / Validierbarkeit	Die Regel ist in der Gegenwartsform als feststehende Tatsache beschrieben ²³ .
Qualitätsanforderungen	Validierbarkeit	Eine Qualitätsanforderung muss realistisch und eindeutig messbar sein (ja/nein).

¹⁹ **Der Ausweis wurde gedruckt...** liegt innerhalb der Systemgrenzen. Die Ergänzung ... **und kann dem Benutzer ausgehändigt werden.** verbessert die Verständlichkeit und ermöglicht das leichtere Prüfen auf Vollständigkeit der Beschreibung, Vorbedingung und der Daten / Funktionen.

²⁰ z.B. **Kein_freier_Sitzplatz_gefunden**, diese dürfen - wenn die Leser des Dokumentes damit keine Probleme haben, auch abgekürzt werden, z.B. **Kein_Sitzplatz** oder auch **Kein_Platz**

²¹ Ausnahmefälle sollten auch übergreifend eindeutige Namen haben, dies wird aber in diesem Checklisten-teil nicht geprüft.

²² Auch für <Zeitpunkt> sollte es Standardsatzteile, feste Schlüsselwörter geben: **Immer, Bis** <Datum>, **30.ter des Monats**

²³ Nicht: **Der Benutzer muss berechtigt sein, ...**, sondern **Der Benutzer ist berechtigt, ...**

Daten / Funktionen	Vollständigkeit	Alle beobachteten Größen sind mit der Kennzeichnung M für monitored variables und alle kontrollierten Größen mit der Kennzeichnung C für controlled variables aufgelistet. Werden Größen sowohl beobachtet als auch kontrolliert, so werden sie ebenfalls als kontrollierte Größe mit C bezeichnet.
--------------------	-----------------	---

Alle Funktionen sind vollständig aufgelistet.

6 Checklisten-basierte Inspektion der Use Cases

Aufbauend auf den Qualitätsmerkmalen und einer Analyse der bisherigen Erfahrungen sieht die Checkliste zur Inspektion von Use Cases in QUASAR so aus:

Nr.	Frage	✓
	Fragen zu dem gesamten Dokument	
1	Existiert zu jedem Use Case im Use Case Diagramm genau ein formell beschriebener Use Case und umgekehrt?	
2	Sind alle Benennungen innerhalb aller Use Cases und innerhalb der einzelnen Use Cases eindeutig (einander nicht zum Verwechseln ähnlich)?	
3	Sind alle Querverweise innerhalb der Use Cases widerspruchsfrei?	
4	Kommen alle beobachteten und kontrollierten Größen des Systemlastenheftes in wenigstens einem Use Case vor?	
5	Ist das gewünschte System durch die Gesamtheit der Use Cases vollständig beschrieben?	
6	Entsprechen die Use Cases den Kundenwünschen?	
	Fragen zu einem einzelnen Use Case	
7	Entspricht ein einzelner Use Case der vorgegebenen Struktur?	
8	Entspricht ein einzelner Use Case den zugehörigen Regeln?	

Nr.	Frage	✓
9	Ist der einzelne Use Case klar und verständlich?	
10	Sind bei allen Interaktionen deutlich Aktion und Reaktion erkennbar?	
11	Unterscheiden sich Name des Use Cases, Ziel und Nachbedingung in Formulierung und Bedeutung klar voneinander?	
12	Sind die Verweise auf andere Use Cases eindeutige <<includes>> und <<extends>> Beziehungen?	
13	Sind die Ziele, Regeln und Qualitätsanforderungen zueinander (innerhalb eines Use Cases und innerhalb aller Use Cases) widerspruchsfrei und realisierbar?	

Sie ist - mit einer anderen Formatierung versehen - in ausreichender Schriftgröße nicht länger als eine DIN A4-Seite, bietet jeweils unterhalb der Fragen Raum für Notizen, enthält auf der rechten Seite eine Spalte zum Abhaken der analysierten Fragen und für die Übersichtlichkeit sorgt die Aufteilung in zwei Blöcke: "Fragen zu dem gesamten Dokument" und "Fragen zu den einzelnen Use Cases" also zu einzelnen Teilen des Dokumentes. Somit sind die äußerlichen Qualitätsmerkmale erfüllt.

Innerhalb von Quasar führten wir eine Fallstudie durch, bei der der erste Entwurf des Systemlastenheftes, der bereits die vollständigen Use Cases enthielt, mit Hilfe einer solchen Checkliste inspiziert wurde. Die Überarbeitung der dabei benutzten Checkliste führte zur im Bericht dargestellten Version.

7 Erstellung von Szenarien

7.1 Perspektivenbasierte Lesetechnik

Die Perspektivenbasierte Lesetechnik (s.a. Lesetechniken und dazu benötigte Dokumente) beruht auf einem Szenario als Hilfsdokument, das für verschiedene Inspektoren unterschiedliche Inhalte hat und daher zu unterschiedlichen Fehlerentdeckungen führen soll (geringe Überlappung). Durch die Gesamtheit aller Szenarien soll eine hohe Überdeckung der im Dokument enthaltenen Fehler erreicht werden.

Eine Möglichkeit solche Szenarien zu erstellen, beruht wie bei der Erstellung der Checklisten auf der Analyse bisher bekannter Fehlerklassen. Fehlerklassenbasiertes Lesen [Porter, Votta, Basili 1995] beinhaltet eine Reihe von Fragen ähnlich einer Checkliste, die sich allerdings nur auf eine (oder wenige) Fehlerklassen beziehen. Somit kann sich der Inspektor während des Lesens auf wenige wichtige Aspekte konzentrieren und für sein Szenario unwichtige Fehler übergehen. Die Fehlersuche wird dadurch intensiver und das Ergebnis laut dem kontrollierten Experiment mit Studierenden von [Porter, Votta, Basili 1995] effektiver.

Eine Variante der Fehlerklassenbasierten Lesetechnik findet sich in [Cheng, Jeffery 1996], die die Einteilung der Szenarien auf der Basis der Function Point Analyse, also der Ein- und Ausgabegrößen, der Dateien und der Enquiries durchführen. Das Ergebnis der Studie zeigte, dass die Inspektoren mit den Function Point-Szenarien weniger Fehler fanden als mit der Ad-hoc-Lesetechnik, wobei sicher die große Erfahrung der Inspektoren einen Einfluss auf das Ergebnis hatte.

[Laitenberger 2000] entwickelte aus den bekannten szenariobasierten Lesetechniken die Perspektivenbasierte Lesetechnik. Sie geht davon aus [Laitenberger, Kohler 2001], dass die Qualität von Software-Dokumenten letztlich durch die Nutzer dieser Dokumente bestimmt wird. Erfüllt das Dokument die Qualitätsanforderungen des jeweiligen Nutzers, so hat es für diesen eine ausreichend gute Qualität. Durch die Konzentration des Inspektors auf seine Perspektive werden nur die Teile des Dokumentes intensiv gelesen, die für den Nutzer interessant sind. In mehreren Experimenten zeigte sich, dass die perspektivenbasierte Lesetechnik insgesamt effektiver ist als die checklistenbasierte [Laitenberger, El Emam, Harbich 2000] und als die ad-hoc Lesetechnik [Laitenberger, DeBaud 1997].

Eine Literaturrecherche ergab nur eine dokumentierte Anleitung zur Erstellung von Szenarien in [Laitenberger, Kohler 2001]. Die Qualität solcher Szenarien

wird nicht für sich bewertet, sondern als hinreichend angenommen, wenn die Inspektion entsprechende Ergebnisse liefert.

Um die Qualität solcher Perspektiven zu erhöhen, wird nachfolgend das Agendakzept eingeführt.

7.2 Agenden

Das Konzept der Agenden beruht auf Arbeiten des FIRST [Winter, Santen, Heisel 1998]. Agenten sollen die Softwareentwickler bei konstruktiven Prozessen unterstützen, indem sie in Tabellenform die Aktivitäten schrittweise beschreiben. Eine solche Agenda hat prinzipiell folgenden Aufbau:

No	Step	Validation
1	Beschreibung des aktuellen Schritts und Darstellung der benötigten Templates	1. Verschiedene Prüfungen, die dem Durchführenden eine sofortige Kontrolle seiner Arbeitsergebnisse ermöglichen.

7.3 Agenda zur Erstellung von Szenarien

No	Step	Validation												
1	<p>Erzeuge eine Dokumentteil/Nutzer-Matrix</p> <p>Template:</p> <table border="1" style="margin-left: 40px;"> <tr> <td>Dokumen tteil</td> <td>Teil 1</td> <td>Teil 2</td> </tr> <tr> <td>Nutzer</td> <td></td> <td></td> </tr> <tr> <td>Nutzerroll e 1</td> <td></td> <td></td> </tr> <tr> <td>Nutzerroll e 2</td> <td></td> <td></td> </tr> </table>	Dokumen tteil	Teil 1	Teil 2	Nutzer			Nutzerroll e 1			Nutzerroll e 2			<ol style="list-style-type: none"> 1. Von jedem Dokument kann eindeutig bestimmt werden, ob es zu diesem Dokumententyp gehört. 2. Allen identifizierten Nutzern lassen sich in einem konkreten Fall reale Personen zuordnen, die die jeweilige Rolle eines Nutzers innehaben. 3. Jeder Abschnitt und jedes Element des Dokumententyps ist den verschiedenen Nutzern zugeordnet.
Dokumen tteil	Teil 1	Teil 2												
Nutzer														
Nutzerroll e 1														
Nutzerroll e 2														
2	<p>Verfasse die Einleitung des Szenarios, indem folgende Fragen beantwortet werden:</p> <ul style="list-style-type: none"> • Wie lautet die Tätigkeitsbezeichnung des Dokumentennutzers? <Rolle> • Welche Aktivität führt er mit dem Dokument durch? <Aktivität> • Was ist ein Erfolgsfaktor bei diesem Teil der Tätigkeit? <Erfolgsfaktor> • Welche Qualitätsaspekte des Dokumentes sind für die Tätigkeit von Bedeutung? <Qualitätsaspekte> <p>Template:</p> <p>Stellen Sie sich vor Sie sind <Nutzer>. Als Teil Ihrer Arbeit müssen Sie <Aktivität>. Dabei ist es für den Erfolg ihrer Tätigkeit besonders wichtig, dass Sie <Erfolgsfaktor> können. Der wesentliche Qualitätsaspekt des Systemlastenheftes ist für Sie <Qualitätsaspekt>.</p>	<ol style="list-style-type: none"> 1. Alle Fragen sind eindeutig beantwortet und als Einleitung formuliert. 2. Für jeden Qualitätsaspekt, der einen anderen Fokus des Inspektors bewirkt, gibt es ein eigenes Szenario. 												
3	<p>Verfasse die Instruktionen des Szenarios in Form einer Agenda.</p>	<ol style="list-style-type: none"> 1. Die Instruktionen geben konkrete Arbeitsaufgaben vor, mit deren Hilfe die Qualität des Dokumentes bezüglich der Informationen aus der Einleitung geprüft werden können. 2. Die Validierung der einzelnen Schritte nimmt Bezug auf typische Probleme, die der Inhaber der Rolle bei der Inspektion entdecken kann. 3. Die Agenda unterstützt vor allem eher in der entsprechenden Rolle unerfahrene Inspektoren. 												
4	<p>Verfasse die offenen Fragen des Szenarios, die typischen Probleme der Adressaten des Dokumentes, wie z.B. Unklarheiten, fehlende Informationen, etc. beinhalten und die die gemachten Erfahrungen bei der Abarbeitung der Aufgaben behandeln.</p>	<ol style="list-style-type: none"> 1. Die Fragen lassen sich nicht mit ja, nein oder quantitativen Angaben beantworten, sondern erfordern komplexere Aussagen. 2. Die Fragen ermöglichen das Reflektieren der Qualitätskriterien des zu inspizierenden Dokumentes. 												

7.4 Beispiel zur Anwendung der Agenda zur Erstellung von Szenarien

Nr	Schritt	Validierung
1	Erzeuge eine Dokumentteil/Nutzer-Matrix	<ol style="list-style-type: none"> 1. Von jedem Dokument kann eindeutig bestimmt werden, ob es zu diesem Dokumententyp gehört. 2. Allen identifizierten Nutzern lassen sich in einem konkreten Fall reale Personen zuordnen, die die jeweilige Rolle innehaben. 3. Jeder Abschnitt und jedes Element des Dokumententyps ist den verschiedenen Nutzern zugeordnet.

Der identifizierte Dokumententyp für das hier benutzte Dokument ist **Systemlastenheft** und daraus speziell der **Abschnitt Use Cases**.

Die potentiellen Nutzer des gesamten Dokumentes sind

- Kunde
- Systempflichtenheftersteller
- Tester
- Projektleitung
- Qualitätsmanagement

Die Dokumentteil /Nutzer-Matrix sieht dann folgendermaßen aus:

Nutzer	Dokumentteil					
	Einleitung	Allg. Beschr.	Use Case Diagramm	Use Cases	Nicht-funkt. Anforderungen	Projektanforderungen
Kunde	(X)	(X)	X	X	X	X
Systempflichtenheftersteller	(X)		X	X	(X)	
Tester	(X)	(X)	X	X	X	
Projektleitung	(X)	(X)	(X)		X	X
Qualitätsmanagement	(X)	(X)			(X)	X

Da es in diesem Bericht um die Inspektion von Use Cases geht, werden die Nutzer Projektleitung und Qualitätsmanagement nicht weiter betrachtet.

Nr	Schritt	Validierung
2	<p>Verfasse die Einleitung des Szenarios, indem folgende Fragen beantwortet werden:</p> <ul style="list-style-type: none"> • Wie lautet die Tätigkeitsbezeichnung des Dokumentennutzers? <Rolle> • Welche Aktivität führt er mit dem Dokument durch? <Aktivität> • Was ist ein Erfolgsfaktor bei diesem Teil der Tätigkeit? <Erfolgsfaktor> • Welche Qualitätsaspekte des Dokumentes sind für die Tätigkeit von Bedeutung? <Qualitätsaspekte> 	1. Alle Fragen sind eindeutig beantwortet und als Einleitung formuliert.

Einleitung:

Stellen Sie sich vor, Sie sind Tester. Als Teil ihrer Arbeit müssen Sie einen Testplan und Testfälle aus dem Systemlastenheft entwickeln. Dabei ist es für den Erfolg ihrer Tätigkeit besonders wichtig, dass Sie Testverfahren zu einzelnen Teilen und die Testfälle ableiten können. Der wesentliche Qualitätsaspekt des Systemlastenheftes ist für Sie die Prüfbarkeit dieser Anforderungen.

Nr	Schritt	Validierung
3	Verfasse die Instruktionen des Szenarios in Form einer Agenda.	<p>1. Die Instruktionen geben konkrete Arbeitsaufgaben vor, mit deren Hilfe die Qualität des Dokumentes bezüglich der Informationen aus der Einleitung geprüft werden können.</p> <p>2. Die Validierung der einzelnen Schritte nimmt Bezug auf typische Probleme, die der Inhaber der Rolle bei der Inspektion entdecken kann.</p> <p>3. Die Agenda unterstützt vor allem eher in der entsprechenden Rolle unerfahrene Inspektoren.</p>

Überlegungen zu der Rolle des Testers:

Der Tester muss folgende Aufgaben durchführen:

- 1.) Den Testplan erstellen, der bestimmt wer was bis wann mit welchen Ressourcen testen wird.
- 2.) Testfälle aufgrund der ihm zur Verfügung stehenden Anforderungsdokumente erarbeiten.
- 3.) Tests mit Hilfe der vorhandenen Testfälle durchführen, dokumentieren und ggfs. Problembereiche erstellen.

Diese Aufgaben sollte er so früh wie möglich beginnen. Dazu wird er zu Beginn des Projektes den Testplan mit Hilfe von frühen Anforderungsanalysedokumenten erstellen und erste Testfälle aufgrund der vorhandenen Dokumente erarbeiten, die er mit zunehmender Zahl an Informationen ergänzen und vervollständigen wird.

Zum Zeitpunkt der Erstellung des Systemlastenheftes ist also lediglich der erste Entwurf eines Testplanes und zu den ausgewählten Dokumentteilen die Erarbeitung exemplarischer Testfälle möglich. Ein wesentlicher Teil der Testplanung ist es, eine Auswahl der benötigten Testverfahren zu treffen.

Erstellung der Agenda:

No	Step	Validation															
1	<p>Erstelle die Testobjekt-Feature-Matrix</p> <ol style="list-style-type: none"> 1. Liste alle Testobjekte bzw. Subsysteme hierarchisch auf, wobei das oberste Testobjekt das gesamte zu testende System ist. 2. Wähle eine Hierarchieebene aus und übertrage diese in die erste Zeile der Testobjekt-Feature-Matrix. 3. Liste in der ersten Spalte alle Features und Qualitätsanforderungen auf, die das System zu erfüllen hat. 4. Notiere in jeder Zelle der Matrix, ob die jeweilige Kombination getestet werden soll. Falls ja, notiere das entsprechende Testverfahren; falls nein, begründe, warum nicht. <p>Template:</p> <table border="1" data-bbox="272 1021 914 1227"> <thead> <tr> <th></th> <th><Test-objekt1></th> <th><Test-objekt2></th> <th><Test-objekt3></th> </tr> </thead> <tbody> <tr> <th><Feature1></th> <td><Test-verfahren oder Grund></td> <td><Test-verfahren oder Grund></td> <td><Test-verfahren oder Grund></td> </tr> <tr> <th><Feature2></th> <td><Test-verfahren oder Grund></td> <td><Test-verfahren oder Grund></td> <td><Test-verfahren oder Grund></td> </tr> </tbody> </table>		<Test-objekt1>	<Test-objekt2>	<Test-objekt3>	<Feature1>	<Test-verfahren oder Grund>	<Test-verfahren oder Grund>	<Test-verfahren oder Grund>	<Feature2>	<Test-verfahren oder Grund>	<Test-verfahren oder Grund>	<Test-verfahren oder Grund>	<ol style="list-style-type: none"> 1. Alle Testobjekte bzw. Subsysteme können in einer hierarchischen Stufe vollständig dem Anforderungsdokument entnommen werden. 2. Alle tatsächlich benötigten Features und Qualitätsanforderungen können dem Anforderungsdokument entnommen werden. 3. Für alle zu testenden Kombinationen lassen sich geeignete Testverfahren aus dem Anforderungsdokument heraus ermitteln. 			
	<Test-objekt1>	<Test-objekt2>	<Test-objekt3>														
<Feature1>	<Test-verfahren oder Grund>	<Test-verfahren oder Grund>	<Test-verfahren oder Grund>														
<Feature2>	<Test-verfahren oder Grund>	<Test-verfahren oder Grund>	<Test-verfahren oder Grund>														
2	<p>Skizziere die Mensch-Maschine-Schnittstelle und notiere den Akteur, mögliche Aktionen des Akteurs, Veranlassung²⁴ dafür und erwarteter Reaktion des Systems entsprechend einer exemplarischen Erstellung von Testfällen.</p> <p>Template:</p> <table border="1" data-bbox="272 1458 895 1592"> <thead> <tr> <th>Punkt auf der Skizze</th> <th>Akteur</th> <th>Aktion</th> <th>Veranlassung / Begründung</th> <th>Reaktion</th> </tr> </thead> <tbody> <tr> <td>(1)</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>(2)</td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Punkt auf der Skizze	Akteur	Aktion	Veranlassung / Begründung	Reaktion	(1)					(2)					<ol style="list-style-type: none"> 1. Die Akteure entsprechen den Akteuren in den Use Cases und im Use Case Diagramm, die Beziehungen sind korrekt. 2. Die Aktionen sind in den Use Cases enthalten. 3. Die Reaktionen sind in den Beschreibungen und den Ausnahmefällen der Use Cases enthalten.
Punkt auf der Skizze	Akteur	Aktion	Veranlassung / Begründung	Reaktion													
(1)																	
(2)																	

²⁴ Veranlassung: Was könnte den Menschen veranlassen, diese Aktion auszuführen? Dies hilft bei dem Autor des Anforderungsdokumentes bei der Überarbeitung.

Nr	Schritt	Validierung
4	Verfasse die offenen Fragen des Szenarios.	1. Die Fragen lassen sich nicht mit ja, nein oder quantitativen Angaben beantworten, sondern erfordern komplexere Aussagen. 2. Die Fragen ermöglichen das Reflektieren der Qualitätskriterien des zu inspizierenden Dokumentes.

Fragen:

1. Welche Features sind unklar?
2. Für welche Zellen in der Testobjekt-Features-Matrix fehlen Ihnen noch Testverfahren?
3. Welche Testfälle sind nicht (vollständig) spezifizierbar?
4. Welche Ereignisse sollten aufgrund ungültiger Aktionen passieren? Wo würden Sie die Ausnahmen ergänzen?
5. Welche Qualitätsanforderungen passen nicht zu Ihrer Skizze der Mensch-Maschine-Schnittstelle?

8 Szenarien für Use Cases in Form von Agenden

Wie bereits in 7.4 Beispiel zur Anwendung der Agenda zur Erstellung von Szenarien dargestellt, gibt wurden für den Dokumenttyp "Systemlastenheft" fünf verschiedene potentielle Nutzer identifiziert:

- Kunde
- Systempflichtenheftersteller
- Tester
- Projektleitung
- Qualitätsmanagement

Davon werden für Projektleitung und Qualitätsmanagement keine Szenarien erarbeitet, da sich deren Nutzer-Interesse wenig bis gar nicht auf das Use Case Diagram und die Use Cases bezieht, was aber Hauptgegenstand der Inspektion sein soll.

Nachfolgend werden nun die Ergebnisse der Szenarien für die ersten drei Nutzer dargestellt. Die Erarbeitung entspricht dem bereits dargestellten Beispiel Tester in 7.4 Beispiel zur Anwendung der Agenda zur Erstellung von Szenarien und wird nicht mehr explizit erläutert.

8.1 Kunde

Stellen Sie sich vor, Sie sind der Kunde des Systemlastenheftes. Als Teil Ihrer Arbeit müssen Sie sicherstellen, dass alle Ihre Anforderungen im Systemlastenheft so dargestellt sind, wie Sie es intendiert haben. Wesentliche Qualitätsaspekte des Systemlastenheftes sind für Sie die Vollständigkeit und die Korrektheit der Anforderungen.

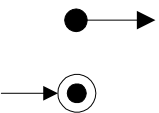

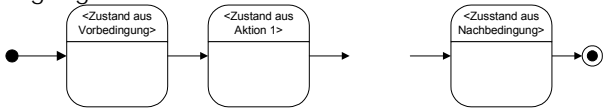
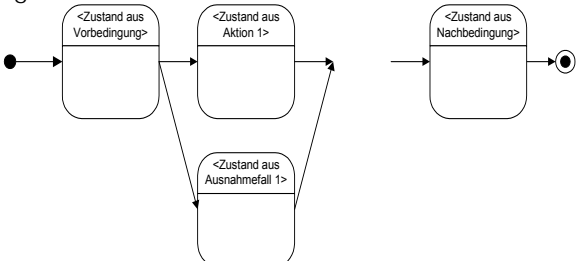
No	Step	Validation
1	Notiere alle konkreten Personen, die das System benutzen werden und fasse diese zu Rollen zusammen. Vergleiche diese Liste mit den Aktoren im Use Case Diagramm und mit den Aktoren, die in den einzelnen Use Cases angegeben werden	1. Alle verschiedenen Rollen kommen als Aktoren im Use Case Diagramm und mit genau gleicher Bezeichnung und Zuordnung zu den einzelnen Use Cases auch in deren textueller Darstellung unter der Rubrik "Aktor" vor.
2	Notiere Sie sich stichpunktartig konkrete Szenarien, die Sie sich zu jedem einzelnen Use Case ausgehend von der Use Case Bezeichnung im Use Case Diagramm vorstellen.	1. Alle Szenarien werden durch Use Cases abgedeckt.
3	Erstelle eine Tabelle, die in der ersten Zeile die Namen aller Use Cases enthält und in der ersten Spalte die Namen aller beobachteten und kontrollierten Größen, die unabhängig vom Anforderungsdokument bei den einzelnen Use Cases Verwendung finden sollten. Ergänze in jeder Zelle dieser Tabelle ein M für beobachtete Größen, ein C für kontrollierte Größen und ein M/C für beobachtete und gleichzeitig kontrollierte Größen. Soll eine Größe in einem Use Case nicht vorkommen, so mache dort einen Strich.	1. Die erzeugte Tabelle korrespondiert mit der Auflistung der beobachteten und kontrollierten Variablen in den einzelnen Use Cases.
4	Prüfe, ob die Qualitätsanforderungen Ihren Anforderungen an das System bezüglich der einzelnen Use Cases entsprechen.	1. Alle Qualitätsanforderungen an das System, die sich auf bestimmte Use Case beziehen, sind dort aufgelistet. 2. Alle aufgelisteten Qualitätsanforderungen in den einzelnen Use Cases sind genau die gewünschten Anforderungen an das System.

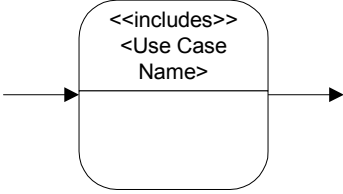
Fragen:

1. Welche Abläufe (insbesondere Ausnahmefälle) können Sie sich vorstellen, die nicht durch den Use Case abgedeckt sind?
2. Wo sind Ihnen Abläufe unklar geblieben?
3. Welche beobachteten oder kontrollierten Größen werden entgegen Ihren Vorstellungen verwendet?
4. Welche benutzten Begriffe sind Ihnen unklar geblieben?
5. Inwieweit wird das Ziel durch den Use Case nicht erreicht?

8.2 Systempflichtenheftersteller

Stellen Sie sich vor, Sie sind Systempflichtenheftersteller. Als Teil ihrer Arbeit müssen Sie einen Überblick über das Systemlastenheft gewinnen. Dabei ist es für den Erfolg ihrer Tätigkeit besonders wichtig, dass Sie einen Zustandsgraphen erarbeiten können. Der wesentliche Qualitätsaspekt des Systemlastenheftes ist für Sie die Vollständigkeit und die Realisierbarkeit einzelner Use Cases.

No	Step	Validation
1	<p>Erstelle die Zustandsgraphen zu den Use Cases. Für jeden Use Case:</p> <p>1. Zeichne den Ausgangszustand und den Endzustand in großem Abstand.</p> <p>2. Suche die Vorbedingung des Use Cases und zeichne dafür einen Zustandsknoten</p> <p>3. Suche die erste Aktion der Beschreibung im Use Case. Zeichne von dem Zustand aus der Vorbedingung einen Zustandsübergang (Pfeil) zum Zustand der Aktion und von dieser zur nächsten Aktion usw. bis zur Nachbedingung</p>	<p>1. Alle Zustände lassen sich aus der Vorbedingung, der Beschreibung und der Nachbedingung entnehmen.</p> <p>2. Alle Zustände können erreicht und wieder verlassen werden.</p> <p>3. Alle möglichen Zustände sind im Use Case beschrieben und im Zustandsgraphen dargestellt.</p> <p>4. Alle möglichen Zustandsübergänge sind im Use Case beschrieben und im Zustandsgraphen dargestellt: Für jeden Zustand ist ersichtlich, ob und unter welchen Bedingungen er betreten, darin verharrt oder wieder verlassen werden kann.</p>
		
		
		
2	<p>Füge für jeden Ausnahmezustand einen Zustand an der geeigneten Stelle hinzu.</p>	<p>1. Alle Zustandsübergänge und Zustände zu den Ausnahmefällen sind im Use Case beschrieben und im Zustandsgraphen dargestellt.</p> <p>2. Alle denkbaren Ausnahmefälle sind im Use Case beschrieben.</p>
		

3	<p>Für alle <<includes>> zeichne einen Zustand, der mit Hinweispfeil und dem Namen des Use Cases beschriftet ist.</p> 	<p>1. Alle <<includes>> Beziehungen sind im Use Case Diagramm und in den Use Cases konsistent und im Zustandsgraphen dargestellt.</p> <p>2. Die referenzierten Use Cases sind im Systemlastenheft enthalten und es existiert ein Zustandsgraph für den inkludierten Use Case.</p>
4	<p>Für jede Regel ...</p>	
5	<p>Für jede monitored / controlled Variable ...</p>	

Fragen:

1. Welche Zustände kommen im Zustandsgraphen, bzw. im Use Case nicht vor, obwohl Sie eintreten könnten?
2. Wo sind Ihnen Abläufe unklar geblieben?
3. Welche Forderungen aus den Use Cases sind nicht oder nur schwierig realisierbar?
4. Welche Ereignisse könnten zusätzlich noch auftreten, die nicht als Ausnahmefälle beschrieben sind?
5. Welche Regeln widersprechen Ihrem Zustandsgraphen und welche fehlen noch?

8.3 Tester

Stellen Sie sich vor, Sie sind Tester. Als Teil ihrer Arbeit müssen Sie einen Testplan und Testfälle aus dem Systemlastenheft entwickeln. Dabei ist es für den Erfolg ihrer Tätigkeit besonders wichtig, dass Sie Testverfahren zu einzelnen Teilen und die Testfälle ableiten können. Der wesentliche Qualitätsaspekt des Systemlastenheftes ist für Sie die Prüfbarkeit dieser Anforderungen.

No	Step	Validation															
1	<p>Erstelle die Testobjekt-Feature-Matrix</p> <ol style="list-style-type: none"> 1. Liste alle Testobjekte bzw. Subsysteme hierarchisch auf, wobei das oberste Testobjekt das gesamte zu testende System ist. 2. Wähle eine Hierarchieebene aus und übertrage diese in die erste Zeile der Testobjekt-Feature-Matrix. 3. Liste in der ersten Spalte alle Features und Qualitätsanforderungen auf, die das System zu erfüllen hat. 4. Notiere in jeder Zelle der Matrix, ob die jeweilige Kombination getestet werden soll. Falls ja, notiere das entsprechende Testverfahren; falls nein, begründe, warum nicht. <p>Template:</p> <table border="1"> <thead> <tr> <th></th> <th><Test-objekt1></th> <th><Test-objekt2></th> <th><Test-objekt3></th> </tr> </thead> <tbody> <tr> <th><Feature1></th> <td><Testverfahren oder Grund></td> <td><Testverfahren oder Grund></td> <td><Testverfahren oder Grund></td> </tr> <tr> <th><Feature2></th> <td><Testverfahren oder Grund></td> <td><Testverfahren oder Grund></td> <td><Testverfahren oder Grund></td> </tr> </tbody> </table>		<Test-objekt1>	<Test-objekt2>	<Test-objekt3>	<Feature1>	<Testverfahren oder Grund>	<Testverfahren oder Grund>	<Testverfahren oder Grund>	<Feature2>	<Testverfahren oder Grund>	<Testverfahren oder Grund>	<Testverfahren oder Grund>	<ol style="list-style-type: none"> 1. Alle Testobjekte bzw. Subsysteme können in einer hierarchischen Stufe vollständig dem Anforderungsdokument entnommen werden. 2. Alle tatsächlich benötigten Features und Qualitätsanforderungen können dem Anforderungsdokument entnommen werden. 3. Für alle zu testenden Kombinationen lassen sich geeignete Testverfahren aus dem Anforderungsdokument heraus ermitteln. 			
	<Test-objekt1>	<Test-objekt2>	<Test-objekt3>														
<Feature1>	<Testverfahren oder Grund>	<Testverfahren oder Grund>	<Testverfahren oder Grund>														
<Feature2>	<Testverfahren oder Grund>	<Testverfahren oder Grund>	<Testverfahren oder Grund>														
2	<p>Skizziere die Mensch-Maschine-Schnittstelle und notiere den Actor, mögliche Aktionen des Aktors, Veranlassung²⁵ dafür und erwarteter Reaktion des Systems entsprechend einer exemplarischen Erstellung von Testfällen.</p> <p>Template:</p> <table border="1"> <thead> <tr> <th>Punkt auf der Skizze</th> <th>Aktor</th> <th>Aktion</th> <th>Veranlassung / Begründung</th> <th>Reaktion</th> </tr> </thead> <tbody> <tr> <td>(1)</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>(2)</td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Punkt auf der Skizze	Aktor	Aktion	Veranlassung / Begründung	Reaktion	(1)					(2)					<ol style="list-style-type: none"> 1. Die Aktoren entsprechen den Aktoren in den Use Cases und im Use Case Diagramm, die Beziehungen sind korrekt. 2. Die Aktionen sind in den Use Cases enthalten. 3. Die Reaktionen sind in den Beschreibungen und den Ausnahmefällen der Use Cases enthalten.
Punkt auf der Skizze	Aktor	Aktion	Veranlassung / Begründung	Reaktion													
(1)																	
(2)																	

²⁵ Veranlassung: Was könnte den Menschen veranlassen, diese Aktion auszuführen? Dies hilft bei dem Autor des Anforderungsdokumentes bei der Überarbeitung.

Fragen:

1. Welche Features sind unklar?
2. Für welche Zellen in der Testobjekt-Features-Matrix fehlen Ihnen noch Testverfahren?
3. Welche Testfälle sind nicht (vollständig) spezifizierbar?
4. Welche Ereignisse sollten aufgrund ungültiger Aktionen passieren? Wo würden Sie die Ausnahmen ergänzen?
5. Welche Qualitätsanforderungen passen nicht zu Ihrer Skizze der Mensch-Maschine-Schnittstelle?

9 Zusammenfassung

Für den dargestellten Inspektionsprozess wurden im QUASAR-Projekt zwei verschiedene Lesetechniken für die Inspektion von Systemlastenheften und insbesondere für die darin enthaltenen Use Cases erarbeitet.

Die dazu benötigten Inspektionshilfsdokumente werden aufbauend auf der Struktur und den geforderten Qualitätsmerkmalen an ein Systemlastenheft erstellt. Dabei werden für die Checklistenbasierte Lesetechnik Fragen zu den einzelnen Qualitätsmerkmalen erarbeitet, blockweise zusammengefasst und den Inspektoren zur Verfügung gestellt.

Für die perspektivenbasierte Lesetechnik wurden ausgehend von den am IESE entwickelten Szenarien und von den von FIRST entwickeltem Agendakzept Agenden zur Inspektion von Use Cases in Systemlastenheften entwickelt. Diese bieten den Vorteil, dass die Inspektoren bereits während der Durchführung der Aufgaben, die zu einer Inspektionsperspektive gehören, prüfende und damit fehleraufdeckende Schritte durchführen und so gezielter zu den Problempunkten des zu inspizierenden Dokumentes kommen.

10 Referenzen

- [Ackerman, Buchwald, Lewsky 1989] Ackerman, A.F.; Buchwald, L.S.; Lewsky, F.H.: Software Inspections: An Effective Verification Process In: IEEE Software, 6. Jg. (1989), H. 3, S. 31-36.
- [Adams 1999] Adams, Tom: A Formula for the Re-Inspection Decision In: ACM SIGSOFT Software Engineering Notes, 24. Jg. (1999), H. 3, S. 80.
- [Anton, Carter, Dagnino 2001] Anton, Annie, I.; Carter, Ryan, A.; Dagnino, Aldo : Deriving Goals from Use-Case Based Requirements Specification. In: ACM SIGSOFT Software Engineering Notes, Jg. 2001, H. 6, S. 63-73.
- [Biffi 2001] Biffi, Stefan: Lesetechniken für die Inspektion von Software-Anforderungsdokumenten Ein kontrolliertes Experiment zur Untersuchung der Effektivität und Verwendung von Zeit bei der Anwendung verschiedener Lesetechniken In: Informatik Forschung und Entwicklung, 16. Jg. (2001), S. 145-158.
- [Briand, Laitenberger, Wieczorek 1997] Brain, Lionel C.; Laitenberger, Oliver; Wieczorek, Isabella: Building Resource and Quality Management Models for Software Inspections International Software Engineering Research Network Technical Report, ISERN-97-06 .
- [Bryczynski 1999] Bryczynski, Bill: A Survey of Software Inspection Checklists In: ACM SIGSOFT Software Engineering Notes, 24. Jg. (1999), H. 1, S. 82-89.
- [Cheng, Jeffery 1996] Cheng, B.; Jeffery, R.: Comparing Inspection Strategies for Software Requirements Specifications Aus: (Hrsg.): Proceedings of the 1996 Australian Software Engineering Conference o.O. 1996. S. 203-211.
- [Chernak 1996] Chernak, Y.: A statistical approach to the inspection checklist formal synthesis and improvement In: IEEE Transactions on Software Engineering, 22. Jg. (1996), H. 12, S. 866-874.
- [Crisione, Ferree, Porter 2001.] Crisione, Mark; Ferree, Jim; Porter, Don: Predicting Software Errors and Defects In: www.stickyminds.com, Jg. 2001, S. 9.

- [Denger, Doerr, Kamsties 2001] Denger, Christian; Doerr, Jörg; Kamsties, Erik: A Survey on Approaches for Writing Precise Natural Language Requirements Fraunhofer Institut Experimentelles Software Engineering, IESE-Report Nr. 070.01/E .
- [Doolan 1992] Doolan, E.P.: Experience with Fagan's Inspection Method In: Software-Practice and Experience, 22. Jg. (1992), H. 3, S. 173-182.
- [Dyer 1992] Dyer, M.: The Cleanroom Approach to Quality Software Development o.O. (John Wiley and Sons, Inc.) 1992.
- [Dyer 1992a] Dyer, M.: Verification-based Inspection Aus: (Hrsg.): Proceedings of the 26th Hawaii International Conference on System Sciences o.O. 1992. S. 418-427.
- [Fagan 1976] M. E. Fagan. Design and Code Inspections to Reduce Errors in Program Development. IBM Systems Journal, 15(3):182-211,1976
- [Fagan 1986] Fagan, Michel E.: Advances in Software Inspections In: IEEE Transactions on Software Engineering, 12. Jg. (1986), H. 7, S. 744-751.
- [Freimut 2001] Freimut, Bernd: Developing and Using Defect Classification Schemes Fraunhofer Institut Experimentelles Software Engineering, IESE-Report Nr. 072.01/E .
- [Gilb, Graham 1993] Gilb, Tom; Graham, Dorothy: Software Inspection o.O. (Addison Wesley Longman Limited) 1993.
- [Graham 2000] Graham, Dorothy: Inspection Myths and Misconceptions Aus: (Hrsg.): Proceedings of the International Conference on Software Testing o.O. u. J.
- [Houdek, Paech 2002] Houdek, Frank; Paech, Barbara 2002: Das Türsteuergerät Eine Beispielspezifikation Fraunhofer Institut Experimentelles Software Engineering, IESE-Report Nr. 002.02/D .
- [IEEE 830] IEEE Std. 830-1993 IEEE Recommend Practice for Software Requirements Specifications, The Institute of Electrical and Electronic Engineers, Inc. New York 1994.
- [Kamsties, Berry, Paech 2001] Kamsties, Erik; Berry, Daniel M.; Paech, Barbara: Detecting Ambiguities in Requirements Documents Using Inspections Aus: (Hrsg.): 1st Workshop on Inspection in Software Engineering. WISE'01 - Proceedings Hamilton 2001. S. 68-80.

- [Kamsties, Knethen, Paech 2001] Kamsties, Erik; Knethen, Antje von; Paech, Barbara: Structure of QUASAR Requirements Documents IESE-Report Nr. 073.01/E.
- [Knethen 2001] Knethen, Antje von: Change-Oriented Requirements Traceability Support for Evolution of Embedded Systems Stuttgart (Fraunhofer IRB Verlag) 2001. (= PhD Theses in Experimental Software Engineering)
- [Knethen, Paech 2002] Knethen, Antje von; Paech, Barbara: A Survey on Tracing Approaches in Practice and Research Kaiserslautern 2002.
- [Laitenberger 2000] O. Laitenberger. Cost-Effective Detection of Software Defects Through Perspective-based Inspections. PhD Theses in Experimental Software Engineering Vol. 1, 2000
- [Laitenberger, Atkinson, Schlich 2000] Laitenberger, Oliver; Atkinson, Colin; Schlich, Maud et al: An Experimental Comparison of Reading Techniques for Defect Detection in UML Design Documents In: Journal of Systems and Software, 53. Jg. (2000), H. 2, S. 183-204.
- [Laitenberger, DeBaud 1997] Laitenberger, Oliver; DeBaud Jean Marc: Perspective-based Reading of Code Documents at Robert Bosch GmbH Fraunhofer Institut Experimentelles Software Engineering, IESE-Report no 049.97/E .
- [Laitenberger, DeBaud 2000] Laitenberger, Oliver; DeBaud, Jean-Marc: An encompassing life cycle centric survey of software inspection In: The Journal of Systems and Software, 50. Jg. (2000), S. 5-31.
- [Laitenberger, El Emam, Harbich 2000] Laitenberger, Oliver; El Emam, Khalid; Harbich, Thomas: An Internally Replicated Quasi-Experimental Comparison of Checklist and Perspective-based Reading of Code Documents In: IEEE Transactions on Software Engineering, Jg. 2000
- [Laitenberger, Kohler 2001] Laitenberger, Oliver; Kohler, Kirstin: The systematic Adaptation of Perspective-based Inspections to Software Development Projects in: Lawford, M.: 1st Workshop on Inspection in Software Engineering, WISE '01; Proceedings; Hamilton : Software Quality Research Lab, 2001; S.105-114 : Ill., Lit.; Conference: Workshop on Inspection in Software Engineering (WISE) <1, 2001, Paris>
- [Linger, Mills, Witt 1979] Linger, R.C.; Mills, H.D.; Witt, B.I.: Structured Programming: Theory and Practice o.O. (Addison-Wesley Publishing Company) 1979.

- [Paech 2002] Vorlesungsskript Requirements Engineering 2002, Universität Kaiserslautern
- [Paech, Santen et al 2001] Paech, Barbara; Santen, Thomas et al: Ziele, Hypothesen und Validierungsmöglichkeiten für das Projekt QUASAR Fraunhofer Institut Experimentelles Software Engineering, IESE-Report Nr. 049.01/D .
- [Parnas, Weiss 1985] Parnas, David L.; Weiss, D.: Active Design Reviews: Principles and Practices Aus: (Hrsg.): Proceedings of the 8th International Conference on Software Engineering o.O. 1985. S. 132-136.
- [Parnas, Weiss 1987] Parnas, David L.; Weiss, D.: Active Design Reviews: Principles and Practice In: Journal of Systems and Software, 7. Jg. (1987), S. 259-265.
- [Porter, Votta, Basili 1995] Porter, A.A.; Votta, L. G.; Basili, Victor R.: Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment In: IEEE Transactions on Software Engineering, 21. Jg. (1995), H. 6, S. 563-576.
- [Rupp 2001] Rupp, Chris: Requirements-Engineering und -Management Professionelle, iterative Anforderungsanalyse für die Praxis o.O. (Hanser Verlag) 2001.
- [Shirey 1992] Shirey, G.C.: How Inspections Fail Aus: (Hrsg.): Proceedings of the 9th International Conference on Testing Computer Software o.O. 1992. S. 151-159.
- [Tervonen 1996] Tervonen, I.: Support for Quality-Based Design and Inspection In: IEEE Software, 13. Jg. (1996), H. 1, S. 44-54.
- [Thelin, Runeson, Regnell 2001] Thelin, Thomas; Runeson, Per; Regnell, Björn: Usage-based reading - an experiment to guide reviewers with use cases In: Information and Software Technology, 43. Jg. (2001), S. 925-938.
- [Weiss, Kimbrough 1995] Weiss, Alan, R.; Kimbrough, Kerry: Requirements Specifications Inspection Checklist and Standard
- [Winter, Santen, Heisel 1998] Kirsten Winter, Thomas Santen, Maritta Heisel: An Agenda for Specifying Software Components with Complex Data Models. Proc. Safecomp '98. W. Ehrenberger (ed.), LNCS, pp. 16-31, Springer-Verlag, 1998.8

Dokumenten Information

Titel: Inspektion des Systemlastenheftes
Datum: Juli 2002
Report: IESE-036.02/D
Status: Final
Klassifikation: Public

Copyright 2002, Fraunhofer IESE.
Alle Rechte vorbehalten. Diese Veröffentlichung darf für kommerzielle Zwecke ohne vorherige schriftliche Erlaubnis des Herausgebers in keiner Weise, auch nicht auszugsweise, insbesondere elektronisch oder mechanisch, als Fotokopie oder als Aufnahme oder sonstwie vervielfältigt, gespeichert oder übertragen werden. Eine schriftliche Genehmigung ist nicht erforderlich für die Vervielfältigung oder Verteilung der Veröffentlichung von bzw. an Personen zu privaten Zwecken.