Softwareprüfungen: Praxisleitfaden für Hersteller und Anwender

Lesekompass

Dieses Whitepaper ist modular aufgebaut und kann je nach Rolle und Fragestellung gezielt gelesen werden:

- **Softwarehersteller** finden in den Kapiteln 3 eine praxisnahe Darstellung, wie Softwareentwicklungs-, Test- und Freigabeprozesse ausgestaltet sein müssen, um eine Prüfung nach IDW PS 880 n.F. zu ermöglichen.
- **Anwenderunternehmen** erhalten insbesondere in Kapitel 2 und Kapitel 4 Orientierung zur Aussagekraft, Abgrenzung und Verwendung von Softwarebescheinigungen im Kontext rechnungslegungs- und kontrollrelevanter Prozesse.
- **Prüfer und prüfungsnahe Funktionen** finden im gesamten Dokument eine zusammenhängende Darstellung der Prüfungssystematik, von der Festlegung geeigneter Kriterien über die Prüfungshandlungen bis zur Berichterstattung und Ableitung der Softwarebescheinigung.

Je nach Fragestellung kann das Dokument vollständig oder selektiv genutzt werden. Die Kapitel sind inhaltlich aufeinander abgestimmt, aber in sich abgeschlossen strukturiert.

Inhalt

Vorwort

Software ist heute wesentlicher Bestandteil nahezu aller Geschäftsprozesse – insbesondere dort, wo Daten verarbeitet, Entscheidungen unterstützt oder Kontrollen automatisiert werden. Mit dieser zunehmenden Bedeutung steigen auch die Anforderungen an Nachvollziehbarkeit, Sicherheit und Ordnungsmäßigkeit der eingesetzten Systeme.

Von Softwareprodukten wird nicht nur funktionale Zuverlässigkeit erwartet. Zunehmend besteht der Anspruch, dass ihre Qualität strukturiert dokumentiert und unabhängig überprüfbar ist.

Gerade kleinere und mittelständische Softwarehersteller stehen hierbei vor besonderen Herausforderungen: Entwicklungsprozesse müssen professionell gestaltet, Tests systematisch durchgeführt und Ergebnisse nachvollziehbar dokumentiert werden, ohne dabei an Flexibilität und Geschwindigkeit zu verlieren.

Gleichzeitig verlangen Anwenderunternehmen, insbesondere aus regulierten Bereichen, zunehmend einen unabhängigen Nachweis über die Ordnungsmäßigkeit der eingesetzten Software.

Softwareprüfungen nach IDW PS 880 n.F. bieten hierfür einen anerkannten Rahmen. Ein Prüfungsbericht nach diesem Standard schafft Transparenz und damit eine belastbare Grundlage für Vertrauen bei Anwenderunternehmen und deren Prüfern.

Dieses Whitepaper ordnet die Anforderungen aus Sicht der Prüfungspraxis ein. Es richtet sich primär an Softwarehersteller, die ihre Entwicklungs-, Test- und Freigabeprozesse prüfbar ausgestalten möchten, sowie an Anwenderunternehmen, die die Aussagekraft von Softwarebescheinigungen sachgerecht bewerten wollen.

Ziel ist eine praxisnahe und strukturierte Darstellung der maßgeblichen Anforderungen.

Wie bereits im Kontext dienstleistungsbezogener IKS gilt auch hier: **Assurance-Berichte ersetzen kein Vertrauen. Sie machen es belegbar.**

R3 Audit & Assurance GmbH
Wirtschaftsprüfungsgesellschaft

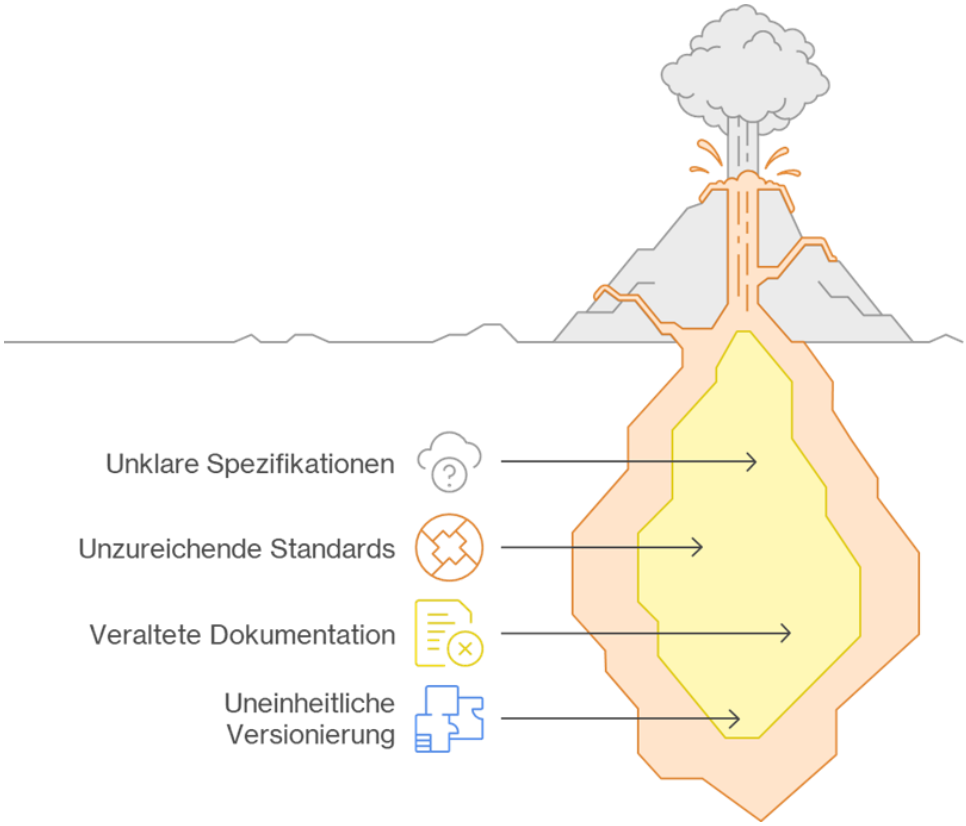
1. Einleitung und Hintergrund

Softwareprüfungen nach IDW PS 880 n.F. werden häufig entweder überschätzt oder unterschätzt. Einige Hersteller befürchten eine technische Detailprüfung des Quellcodes oder eine nahezu vollständige Durchleuchtung ihrer Entwicklungsorganisation. Andere betrachten die Prüfung als rein formalen Nachweis, der sich im Wesentlichen auf die Bestätigung vorhandener Testdokumente beschränkt.

Beides greift zu kurz.

Eine Softwareprüfung ist weder eine Code-Revision noch eine reine Dokumentationsprüfung. Sie ist eine risikoorientierte, kriteriensbezogene Beurteilung eines konkret abgegrenzten Softwareprodukts in einem bestimmten Versionsstand. Im Mittelpunkt steht die Frage, ob Softwaredesign, Entwicklung und Testverfahren geeignet sind, die definierten fachlichen und regulatorischen Anforderungen ordnungsgemäß umzusetzen.

In der Praxis scheitern Softwareprüfungen selten an einzelnen Programmfehlern. Sie scheitern an strukturellen Schwächen:



Diese Schwächen sind nicht zwangsläufig Ausdruck mangelnder Qualität. Häufig resultieren sie aus gewachsenen Strukturen, pragmatischen Vorgehensweisen oder einem unzureichenden Verständnis der prüferischen Perspektive.

Dieses Whitepaper verfolgt daher zwei Ziele:

1. Es erläutert präzise, was eine Softwareprüfung nach IDW PS 880 n.F. tatsächlich beurteilt und was nicht.
2. Es zeigt aus prüferischer Sicht, wie ein Entwicklungs- und Testprozess ausgestaltet sein muss, um belastbar zu sein.

Ziel ist keine theoretische Darstellung des Standards, sondern eine systematische Einordnung der entscheidenden Risiko- und Erfolgsfaktoren aus der Prüfungspraxis.

2. Was eine Softwareprüfung tatsächlich beurteilt

Eine Softwareprüfung nach IDW PS 880 n.F. ist eine produktbezogene, risikoorientierte Beurteilung eines klar abgegrenzten Softwareprodukts in einem definierten Versionsstand.

Sie beantwortet eine zentrale Frage:

Ist das geprüfte Softwareprodukt bei sachgerechter Anwendung geeignet, die festgelegten Kriterien ordnungsgemäß zu erfüllen?

Drei Elemente bestimmen jede Prüfung:

1. der klar definierte Prüfungsgegenstand
2. geeignete Kriterien
3. eine risikoorientierte Prüfungsdurchführung

Prüfung bedeutet Abgrenzung. Ohne klare Abgrenzung kein belastbares Urteil.

Alles Weitere folgt daraus.

2.1 Prüfungsgegenstand – Klarheit vor Beginn der Prüfung

Prüfungsgegenstand ist immer ein konkret definierter Versions- oder Releasestand eines Softwareprodukts oder eines eindeutig abgegrenzten Funktionsbereichs.

Entscheidend ist nicht der gesamte Entwicklungsprozess des Unternehmens, sondern die Funktionsweise der geprüften Software in dem vereinbarten Umfang.

Nicht Prüfungsgegenstand sind:

- der laufende Betrieb bei einzelnen Anwendern,
- allgemeine Organisationsprozesse,
- oder nicht einbezogene Module.

Unklare Abgrenzungen gehören zu den häufigsten Ursachen späterer Missverständnisse im Prüfungsverlauf.

2.2 Kriterien – Der Maßstab der Beurteilung

Eine Softwareprüfung beurteilt nicht „Qualität“ im abstrakten Sinn. Maßstab sind definierte Kriterien.

Diese können sich ergeben aus:

- gesetzlichen oder regulatorischen Anforderungen,
- allgemein anerkannten Grundsätzen (z. B. im Rechnungslegungsumfeld),
- oder vom Hersteller selbst entwickelten, sachgerechten Kriterien.

Entscheidend ist, dass die Kriterien:

- relevant,
- vollständig,
- nachvollziehbar
- und neutral formuliert sind.

Ohne geeignete Kriterien ist keine belastbare Prüfung möglich.

2.3 Was konkret beurteilt wird

Die Prüfung umfasst zwei Dimensionen:

1. Angemessenheit des Designs

Ist die Software konzeptionell geeignet, die definierten Kriterien zu erfüllen?

2. Ordnungsmäßigkeit und Funktionsfähigkeit

Werden die definierten Kriterien im geprüften Versionsstand tatsächlich erfüllt?

Die Beurteilung erfolgt risikoorientiert, stichprobenweise und auf Basis geeigneter Nachweise. Eine vollständige Prüfung sämtlicher Programmfunktionen ist weder vorgesehen noch erforderlich.

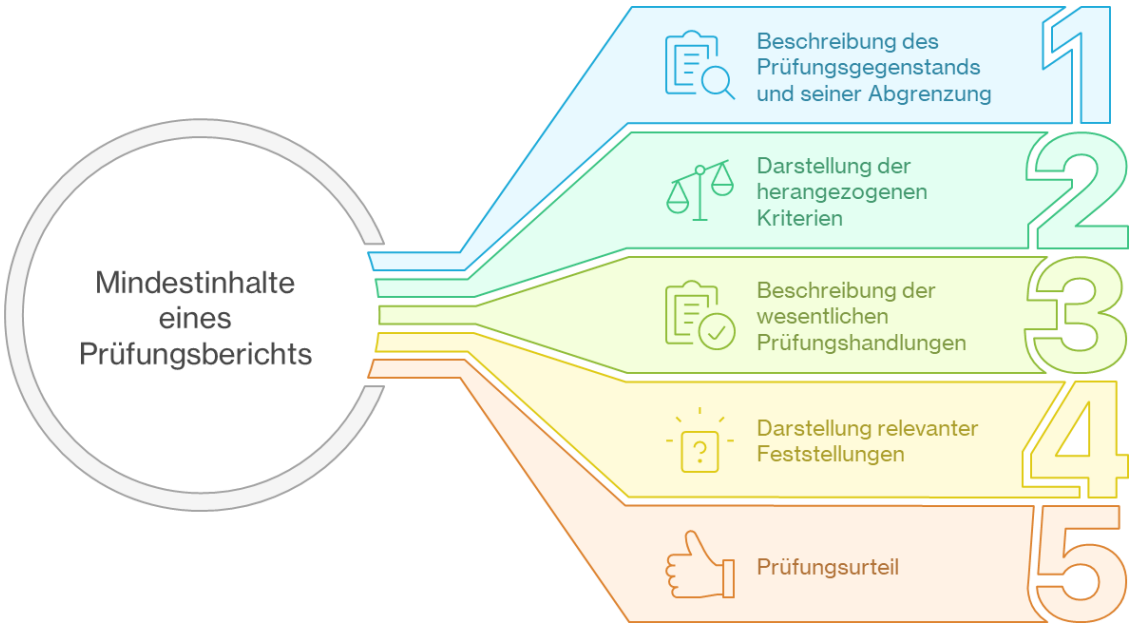
2.4 Prüfungsurteil und Softwarebescheinigung

Das Ergebnis einer Softwareprüfung wird in einem Prüfungsbericht dokumentiert. Dieser stellt die zentrale Grundlage der Beurteilung dar.

Auf Basis des Prüfungsberichts kann eine Softwarebescheinigung erteilt werden. Die Bescheinigung fasst das Prüfungsurteil in komprimierter Form zusammen und verweist auf den zugrunde liegenden Prüfungsbericht.

Während der Prüfungsbericht die methodische Herleitung und die wesentlichen Feststellungen enthält, dient die Softwarebescheinigung in der Praxis häufig als Nachweis gegenüber Anwendern, deren Abschlussprüfern oder weiteren Stakeholdern. Hierbei ist zu beachten, dass die Aussagekraft der Softwarebescheinigung ohne den Prüfungsbericht nur sehr begrenzt ist.

Die Mindestbestandteile eines Prüfungsberichts lassen sich wie folgt strukturieren:



1. Beschreibung des Prüfungsgegenstands und seiner Abgrenzung

Es wird klar definiert, welche Software, welcher Funktionsumfang und welcher Versionsstand Gegenstand der Prüfung waren.

2. Darstellung der herangezogenen Kriterien

Der Bericht legt offen, anhand welcher Maßstäbe die Software beurteilt wurde.

3. Beschreibung der wesentlichen Prüfungshandlungen

Es wird erläutert, wie der Prüfer vorgegangen ist (z. B. Stichproben, Funktionstests, Dokumentenprüfungen).

4. Darstellung relevanter Feststellungen

Wesentliche Abweichungen oder Einschränkungen werden transparent gemacht.

5. Prüfungsurteil

Abschließend wird das Gesamturteil zur Ordnungsmäßigkeit des geprüften Softwareprodukts formuliert.

Abgrenzung und Erwartungsmanagement

Wichtig ist: Eine Softwareprüfung bestätigt nicht die absolute Fehlerfreiheit eines Produkts. Der Prüfungsbericht dokumentiert vielmehr eine risikoorientierte, kriteriensbezogene Beurteilung auf Basis geeigneter Prüfungshandlungen.

Missverständnisse entstehen häufig, wenn Umfang und Aussagekraft eines Prüfungsberichts überschätzt oder mit anderen Prüfungsformaten gleichgesetzt werden. Die transparente Strukturierung der Berichtsinhalte trägt daher wesentlich zur sachgerechten Einordnung bei.

2.5 Abgrenzung zu verwandten Standards

Softwareprüfungen nach IDW PS 880 n.F. werden in der Praxis häufig mit anderen Prüfungs- und Zertifizierungsformaten gleichgesetzt. Dies führt regelmäßig zu Fehlinterpretationen über Reichweite und Aussagekraft der jeweiligen Berichte.

Der wesentliche Unterschied liegt im Prüfungsgegenstand: IDW PS 880 n.F. ist konsequent produktbezogen. Geprüft wird ein konkret abgegrenztes Softwareprodukt in einem definierten Versionsstand.

Andere Prüfungsformen verfolgen eine andere Zielrichtung, wie die folgende Übersicht zeigt:

Merkmal	**IDW PS 880 n.F.**	**IDW PS 951 n.F. / ISAE 3402**	**Abschluss-prüfung**	**ISO-/ Management-systemprüfung**
Prüfungs-gegenstand	Konkretes Softwareprodukt in definiertem Versionsstand	Dienstleistungs-bezogenes internes Kontrollsystem	Rechnungslegung eines Unternehmens	Organisations- oder Management-system
Fokus	Funktionsfähigkeit und Ordnungsmäßig-keit der Software	Ausgestaltung und ggf. Wirksamkeit von Kontrollen	Ordnungsmäßig-keit des Jahres-abschlusses	Einhaltung definierter Prozess- und Governance-Vorgaben
Zeitbezug	Stichtags- bzw. versionsbezogen	Zeitraumbezogen (z. B. 6–12 Monate)	Geschäftsjahresbe zogen	Auditzeitraum (regelmäßig wiederkehrend)
Technischer Funktions-nachweis	Ja (funktionale Prüfung des Produkts)	Nein	Nur insoweit relevant für Rechnungslegung	Nein
Organisations-prüfung	Nur mittelbar, soweit für Produkt relevant	Ja	Ja (im Kontext der Rechnungslegung)	Ja
Aussage über konkrete Softwareversion	Ja	Nein	Nein	Nein
Typische Zielgruppe des Berichts	Softwarehersteller, Anwender, deren Prüfer	Auslagernde Unternehmen und deren Prüfer	Gesellschafter, Investoren, Banken	Kunden, Geschäfts-partner

Die Unterschiede betreffen nicht das Anspruchsniveau, sondern den Fokus der jeweiligen Prüfung.

- IKS-Prüfungen beurteilen Kontrollumfelder und Dienstleistungsprozesse.
- Abschlussprüfungen beurteilen unternehmensspezifische Rechnungslegungsrisiken.
- ISO-Zertifizierungen beurteilen organisatorische Strukturen und Governance-Mechanismen.
- IDW PS 880 n.F. beurteilt die funktionale Ordnungsmäßigkeit eines konkreten Softwareprodukts.

In der Praxis entstehen Missverständnisse häufig daraus, dass Prüfungsberichte mit unterschiedlichem Prüfungsgegenstand funktional gleichgesetzt werden. Eine sachgerechte Einordnung setzt daher ein präzises Verständnis des jeweiligen Prüfungsziels voraus.

3. Der prüfbare Softwareentwicklungsprozess

Eine Softwareprodukt basiert stets auf einem belastbaren und nachvollziehbaren Entwicklungsprozess.

IDW PS 880 n.F. schreibt kein bestimmtes Entwicklungsmodell vor. Weder klassische Wasserfallmodelle noch agile Methoden sind per se vorzugswürdig oder kritisch. Maßgeblich ist allein, ob der gewählte Ansatz geeignet ist, die Einhaltung der definierten Kriterien systematisch sicherzustellen und für einen sachverständigen Dritten nachvollziehbar zu dokumentieren.

Für die Prüfung ist daher nicht entscheidend, *wie* Software entwickelt wird, sondern ob:

- Anforderungen klar definiert sind,
- Design und Implementierung strukturiert erfolgen,
- Tests systematisch durchgeführt und dokumentiert werden,
- Änderungen kontrolliert erfolgen,
- und der geprüfte Versionsstand eindeutig abgegrenzt ist.

Gute Software ist kein Zufallsprodukt, sondern das Ergebnis eines strukturierten Entwicklungsprozesses

Prüfbarkeit entsteht durch Konsistenz, nicht durch Methodendogmatik.

In der Prüfungspraxis zeigt sich regelmäßig: Nicht fehlende Formalisierung ist das Problem, sondern fehlende Durchgängigkeit.

3.1 Anforderungen: Ausgangspunkt jeder prüfbaren Entwicklung

Jeder prüfbare Entwicklungsprozess beginnt mit klar definierten Anforderungen.

Aus prüferischer Sicht sind die Anforderungen an die Software zentral, weil sich die Angemessenheit von Design, Implementierung und Tests ausschließlich im Verhältnis zu den definierten Kriterien beurteilen lässt. Ohne klar dokumentierte Anforderungen ist eine belastbare Beurteilung nicht möglich.

Dabei ist nicht entscheidend, in welchem Format Anforderungen beschrieben werden. User Stories, Ticketsysteme, Lasten-/Pflichtenhefte oder strukturierte Backlogs können gleichermaßen geeignet sein, sofern sie eindeutig, nachvollziehbar und ausreichend konkret formuliert sind. Was nicht als Anforderung beschrieben ist, kann weder systematisch umgesetzt noch belastbar getestet werden.

3.2 Design und Implementierung: Angemessenheit der Umsetzung

Das Softwaredesign bildet die Brücke zwischen Anforderungen und technischer Umsetzung. In dieser Phase wird festgelegt, wie fachliche und technische Anforderungen strukturell umgesetzt werden sollen.

Für die Prüfung stellt sich hier die Frage: Ist das gewählte Design geeignet, die definierten Anforderungen ordnungsgemäß abzubilden?

Ein prüfbares Design muss daher erkennen lassen:

- wie Anforderungen in Module oder Komponenten überführt wurden,
- wie zentrale Verarbeitungslogiken aufgebaut sind,
- welche Annahmen zugrunde liegen,
- und welche Abhängigkeiten bestehen.

Die Implementierung wiederum setzt diese Struktur technisch um.

Prüfungsrelevant ist dabei nicht der Quellcode als solcher, sondern:

- ob Abweichungen vom Design nachvollziehbar sind,
- ob Änderungen dokumentiert wurden,
- und ob die Umsetzung geeignet ist, die Kriterien zu erfüllen.

3.3 Test & Qualitätssicherung – der zentrale Nachweis

Tests und deren Dokumentation sind von zentraler Bedeutung bei einer Softwareprüfung. Während Anforderungen, Design und Implementierung die Grundlage bilden, entscheidet sich im Testprozess, ob der geprüfte Versionsstand die definierten Kriterien tatsächlich erfüllt.

In der Praxis ist dabei weniger die technische Komplexität ausschlaggebend als die Struktur und Nachvollziehbarkeit der Testnachweise.

3.3.1 Prüfungsrelevante Grundsätze für Testverfahren

Ein prüfbarer Testprozess muss nicht komplex sein. Er muss jedoch:

- auf klar definierten Anforderungen beruhen,
- risikoorientiert ausgestaltet sein,
- die wesentlichen Funktionen abdecken,
- nachvollziehbar dokumentiert sein,

- und eine eindeutige Versionenzuordnung ermöglichen.

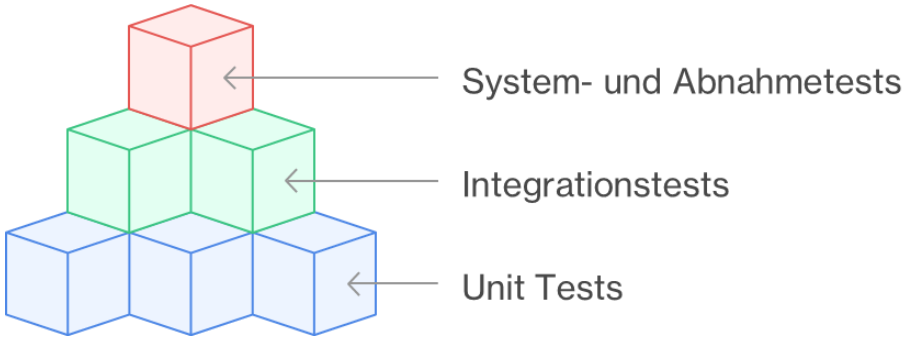
Dabei gilt ein Grundsatz, der in der Prüfungspraxis häufig unterschätzt wird: **Nicht dokumentierte Tests gelten als nicht durchgeführt.**

Eine hohe Anzahl an Testfällen ersetzt keine strukturierte Teststrategie. Entscheidend ist nicht die Quantität, sondern die risikoorientierte Abdeckung der maßgeblichen Anforderungen.

3.3.2 Teststufen – wann wird geprüft?

Ein prüfbarer Testprozess folgt typischerweise einer abgestuften Struktur. Die einzelnen Teststufen bauen inhaltlich aufeinander auf und erfüllen unterschiedliche Funktionen im Gesamtgefüge der Qualitätssicherung.

Die nachfolgende Darstellung verdeutlicht das typische Zusammenspiel:



Die Aussagekraft einer Softwareprüfung steigt mit zunehmender Testtiefe und -Nähe zum realen Nutzungskontext. Gleichzeitig setzen belastbare End-to-End-Tests stabile Modul- und Integrationsprüfungen voraus.

Unit Tests: Stabilität auf Modulebene

Unit Tests prüfen einzelne Funktionen oder Module isoliert. Sie fokussieren sich auf die fachliche Logik und technische Korrektheit.

Aus prüferischer Sicht bilden sie die erste Verteidigungslinie gegen logische Fehlfunktionen. Besonders relevant sind sie dort, wo zentrale Berechnungslogiken, Validierungen oder Entscheidungsregeln umgesetzt werden.

Eine vollständige Testabdeckung ist nicht erforderlich. Entscheidend ist vielmehr, dass kritische Funktionen gezielt abgesichert sind und automatisiert reproduzierbar getestet werden können.

Unit Tests erhöhen die Stabilität des Systems und bilden häufig die Grundlage für wirksame Regressionstests bei späteren Änderungen.

Für kleine Teams sind automatisierte Unittests besonders wertvoll, weil sie ohne manuellen Aufwand wiederholbar sind.

Integrationstests: Kontrolle des Zusammenspiels

Integrationstests prüfen das Zusammenwirken mehrerer Module oder externer Schnittstellen.

Erfahrungsgemäß entstehen Fehler nicht nur innerhalb einzelner Komponenten, sondern vor allem an deren Übergängen, etwa bei Datenformaten, Schnittstellenparametern oder Verarbeitungsreihenfolgen.

Für die Softwareprüfung sind Integrationstests deshalb besonders relevant, wenn:

- externe Komponenten eingebunden sind,
- mehrere Verarbeitungsschritte ineinandergreifen,
- oder Schnittstellen rechnungslegungs- oder kontrollrelevant sind.

Sie schaffen Transparenz darüber, ob die Software nicht nur isoliert funktioniert, sondern im Zusammenspiel konsistent arbeitet.

System- und Abnahmetests: End-to-End-Nachweis der Funktionsfähigkeit

System- und Abnahmetests bilden gemeinsam die höchste Teststufe.

Während Systemtests die technische Funktionsfähigkeit vollständiger Prozessketten unter realistischen Bedingungen prüfen, validieren Abnahmetests (User Acceptance Tests) die fachliche Eignung aus Anwendersicht.

Für eine Softwareprüfung nach IDW PS 880 n.F. sind beide Testformen von hoher Aussagekraft, da sie dokumentieren, dass:

- vollständige Geschäftsprozesse durchgängig funktionieren,
- fachliche Anforderungen nachvollziehbar umgesetzt wurden,
- und das Softwareprodukt im vorgesehenen Nutzungskontext belastbar einsetzbar ist.

Strukturiert dokumentierte User Acceptance Test-Protokolle sind besonders wertvoll, da sie eine externe fachliche Validierung darstellen. Sie ersetzen jedoch keine systematische Testplanung des Herstellers, sondern ergänzen diese.

Einordnung aus Prüfersicht

In der Prüfungspraxis zeigt sich regelmäßig:

- Fehlende Unit- oder Integrationstests führen häufig zu erhöhtem Bedarf an eigenen Funktionstests durch den Prüfer.
- Gut dokumentierte System- und Abnahmetests reduzieren dagegen regelmäßig den Umfang zusätzlicher Prüfungshandlungen.
- Eine ausgewogene Teststruktur über alle Ebenen hinweg erhöht die Effizienz der Prüfung erheblich.

Entscheidend ist nicht die formale Existenz aller Teststufen, sondern die nachvollziehbare risikoorientierte Abdeckung wesentlicher Funktionen.

3.3.3 Testarten – was wird geprüft?

Während Teststufen den zeitlichen und strukturellen Ablauf bestimmen, beschreiben Testarten die inhaltliche Ausrichtung.

Ein belastbares Testkonzept kombiniert mehrere Testarten.

Funktionale Tests

Funktionale Tests prüfen, ob die Software definierte fachliche Anforderungen korrekt umsetzt.

Für eine Softwareprüfung ist wesentlich, dass erkennbar ist:

- welche Anforderung getestet wurde,
- welches Ergebnis erwartet wurde,
- welches Ergebnis tatsächlich eintrat.

Fehlende Erwartungsdefinitionen oder unklare Sollwerte mindern die Aussagekraft erheblich.

Nicht-funktionale Tests

Nicht-funktionale Tests betreffen Eigenschaften wie Performance, Stabilität oder Fehlerverhalten.

Sie sind insbesondere dann prüfungsrelevant, wenn:

- Performance Einfluss auf die Ordnungsmäßigkeit von Prozessen hat,
- Systemabstürze oder Instabilität zu Datenverlust führen können,
- Fehlermeldungen Einfluss auf Nutzerentscheidungen haben.

Ein kurzer, gezielt eingesetzter Performancetest kann oft ausreichen. Entscheidend ist die nachvollziehbare Risikobetrachtung.

Sicherheitstests

Sicherheitstests müssen dem Risikoprofil des Produkts angemessen sein.

Für kleinere Hersteller genügt regelmäßig ein pragmatischer Ansatz, etwa:

- Überprüfung von Rollen- und Berechtigungskonzepten
- Einsatz eines Dependency-Scanners
- grundlegende statische Codeanalysen

Prüfungsrelevant ist weniger die Tiefe einzelner Tools, sondern die dokumentierte Auseinandersetzung mit sicherheitsrelevanten Risiken.

Regressionstests

Regressionstests stellen sicher, dass Änderungen an der Software keine bestehenden Funktionen unbeabsichtigt beeinträchtigen.

Sie werden typischerweise ausgelöst durch:

- Codeänderungen oder neue Features
- Fehlerbehebungen
- Refactorings
- Updates externer Bibliotheken oder Komponenten
- Anpassungen von Schnittstellen

Technisch bestehen Regressionstests in der erneuten Ausführung bereits definierter Testfälle oder Testsets, die zuvor erfolgreich durchlaufen wurden. Ziel ist es, festzustellen, ob bekannte und bereits geprüfte Funktionen weiterhin korrekt arbeiten.

In der Praxis erfolgen Regressionstests häufig:

- automatisiert im Rahmen von Unit- oder Integrationstest-Suiten
- vor jedem Release
- nach sicherheitsrelevanten Updates
- oder nach strukturellen Änderungen am Code

Aus prüferischer Sicht sind Regressionstests besonders bedeutsam, weil sie dokumentieren, dass Änderungen kontrolliert erfolgen und die Stabilität des Systems nicht dem Zufall überlassen wird.

Fehlende oder informell durchgeführte Regressionstests sind eine der häufigsten Ursachen für die Notwendigkeit zusätzlicher Prüfungshandlungen, insbesondere wenn sich nicht nachvollziehen lässt, welche Auswirkungen Änderungen auf bestehende Prozesse hatten.

Explorative Tests

Explorative Tests sind erfahrungsbasierte, nicht vollständig vordefinierte Testdurchläufe.

Im Gegensatz zu strukturierten Testfällen mit klar definierten Eingaben und erwarteten Ergebnissen verfolgt der Tester hier einen offenen Ansatz: Er nutzt die Software aktiv, kombiniert unterschiedliche Eingaben, variiert Abläufe und prüft bewusst ungewöhnliche oder grenzwertige Szenarien.

Explorative Tests sind insbesondere sinnvoll:

- bei neuen oder komplexen Funktionen
- bei vielschichtigen Benutzeroberflächen
- wenn Nutzerflüsse schwer vollständig formalisiert werden können
- zur Identifikation unerwarteter Wechselwirkungen

Typischerweise werden explorative Tests dokumentiert durch:

- eine Beschreibung des getesteten Szenarios
- die Dauer des Tests
- identifizierte Auffälligkeiten oder Fehler
- ggf. Screenshots oder Logauszüge

Sie erhöhen die Testtiefe erheblich, sind jedoch kein Ersatz für strukturierte Testfälle. Für eine Softwareprüfung ist entscheidend, dass zumindest nachvollziehbar dokumentiert ist, in

welchem Umfang explorative Tests durchgeführt wurden und welche Ergebnisse sie erbracht haben.

Zusammenspiel von Teststufen und Testarten

Für eine belastbare Testabdeckung ist nicht entscheidend, ob jede Testart auf jeder Teststufe formal vorhanden ist. Maßgeblich ist vielmehr, dass die gewählten Testarten geeignet sind, die auf der jeweiligen Ebene bestehenden Risiken angemessen abzudecken.

Unit-Tests adressieren primär logische Einzelrisiken. Integrationstests fokussieren Schnittstellen- und Zusammenspielrisiken. System- und Abnahmetests prüfen vollständige Geschäftsprozesse im realitätsnahen Kontext.

Regressionstests durchziehen alle Ebenen. Ihre Bedeutung nimmt jedoch mit steigender Prozessnähe zu: Während Regression auf Modulebene die technische Stabilität absichert, sichern höherstufige Regressionstests die Funktionsfähigkeit gesamter Geschäftsprozesse nach Änderungen.

Für die Prüfung nach IDW PS 880 n.F. ist entscheidend, dass dieses Zusammenspiel nachvollziehbar dokumentiert ist und sich die Teststrategie erkennbar an den wesentlichen Risiken orientiert.

3.3.4 Testmanagement – Nachvollziehbarkeit als Schlüssel

Ein wirksames Testmanagement stellt sicher, dass Tests nicht zufällig oder rein situativ erfolgen, sondern geplant, nachvollziehbar durchgeführt und konsistent dokumentiert werden.

Für eine Softwareprüfung ist dabei nicht entscheidend, wie formal oder umfangreich die Testdokumentation ausgestaltet ist. Entscheidend ist, dass ein sachverständiger Dritter erkennen kann:

- welche Anforderungen getestet wurden,
- mit welcher Methode,
- mit welchem Ergebnis,
- und auf welchen Versionsstand sich die Tests beziehen.

In der Prüfungspraxis zeigen sich fünf Kernbereiche eines belastbaren Testmanagements.

1. Testplanung

Die Testplanung definiert den Rahmen der Testaktivitäten. Sie muss erkennen lassen:

- welche fachlichen und technischen Anforderungen adressiert werden,
- welche Module, Schnittstellen oder Prozesse getestet werden,
- in welchem Umfang getestet wird,
- und wer für Durchführung und Freigabe verantwortlich ist.

2. Strukturierte Testfälle

Testfälle bilden das operative Instrument der Qualitätssicherung.

Ein prüfbarer Testfall sollte zumindest enthalten:

- Ziel oder Zweck des Tests,
- definierte Eingaben oder Rahmenbedingungen,
- das erwartete Ergebnis,
- das tatsächlich eingetretene Ergebnis,
- sowie die Behandlung festgestellter Abweichungen.

Testfälle müssen nicht umfangreich sein. Sie müssen reproduzierbar und eindeutig sein. Unklare Sollwerte oder fehlende Erwartungsdefinitionen reduzieren die Aussagekraft erheblich.

3. Testabdeckung

Aus der Gesamtdokumentation muss nachvollziehbar sein, dass alle wesentlichen Anforderungen mindestens einmal getestet wurden.

Dies erfolgt typischerweise durch:

- die Zuordnung von Testfällen zu Anforderungen,
- eine Prozessübersicht über getestete Geschäftsabläufe,
- oder eine einfache Testmatrix.

Nicht die formale Vollständigkeit ist entscheidend, sondern die risikoorientierte Abdeckung relevanter Funktionen.

4. Dokumentation der Testergebnisse

Die Dokumentation ist der eigentliche Prüfungsnachweis.

Erforderlich ist insbesondere:

- eine eindeutige Kennzeichnung von Testergebnissen (bestanden/nicht bestanden),
- objektive Nachweise wie Screenshots, Logs oder Ergebnisdateien,
- dokumentierte Fehlerbehandlungen,
- sowie Wiederholungstests nach Korrekturen.

5. Freigabe und Versionsbezug

Vor Auslieferung eines Softwarestandes muss eine dokumentierte Freigabe erfolgen.

Dabei muss erkennbar sein:

- welche Tests abgeschlossen wurden,
- welche Abweichungen ggf. noch bestehen,
- wie diese bewertet wurden,
- und auf welchen konkreten Versions- oder Releasestand sich die Freigabe bezieht.

Gerade in kleineren Teams kann dies pragmatisch erfolgen, etwa über ein Freigabeprotokoll oder einen dokumentierten Release-Kommentar im Ticketsystem. Entscheidend ist die eindeutige Nachvollziehbarkeit, nicht die formale Ausgestaltung.

Einordnung aus Prüfersicht

In der Praxis führen nicht fehlende Testarten, sondern fehlende Struktur und fehlender Versionsbezug zu Prüfungshemmnissen.

Ein klar strukturiertes, konsistent dokumentiertes Testmanagement reduziert regelmäßig den Umfang ergänzender Prüfungshandlungen erheblich.

3.3.5 Automatisierung und CI/CD

Automatisierte Tests und Build-Pipelines erhöhen die Reproduzierbarkeit und Stabilität.

Für die Prüfbarkeit sind insbesondere relevant:

- dokumentierte Build-Prozesse
- reproduzierbare Versionsstände
- automatisierte Ausführung definierter Testsets
- protokollierte Testergebnisse

Automatisierung ist kein Selbstzweck. Sie ist dort sinnvoll, wo sie konsistente und nachvollziehbare Ergebnisse liefert. Ein klar definierter manueller Prozess kann ebenso prüfbar sein, sofern er strukturiert dokumentiert ist.

3.4 Release-, Change- und Configuration-Management

Ein strukturiertes Release-, Change- und Configuration-Management stellt sicher, dass Änderungen am Softwareprodukt kontrolliert erfolgen, nachvollziehbar dokumentiert sind und konsistent in Entwicklungs-, Test- und Produktionsumgebungen überführt werden.

Für die Softwareprüfung ist dieser Bereich zentral, weil der Prüfer beurteilen muss, **welcher konkrete Versionsstand geprüft wurde**, **welche Änderungen enthalten sind** und **ob Testergebnisse und Freigaben eindeutig diesem Stand zugeordnet werden können**. Fehlt diese Nachvollziehbarkeit, verliert selbst eine gute Testdokumentation an Aussagekraft.

3.4.1 Change Management

Änderungen am Softwareprodukt (Fehlerbehebungen, neue Funktionen, technische Anpassungen) müssen systematisch erfasst, bewertet und gesteuert werden. Entscheidend ist nicht die formale Ausgestaltung, sondern die Transparenz der Änderungskette:

- **Auslöser und Ziel der Änderung:** Warum wird geändert? Welche Anforderung oder welches Problem wird adressiert?
- **Umsetzung und Nachvollziehbarkeit:** Welche Änderungen wurden konkret umgesetzt (z. B. Tickets, Commits, Pull Requests)?
- **Prüfung der Auswirkungen:** Wurden die Auswirkungen der Änderung bewertet (fachlich/technisch)?

- **Testnachweise nach Änderungen:** Wurden angemessene Regressionstests durchgeführt und dokumentiert?
- **Freigabe vor Integration/Release:** Wer hat die Änderung freigegeben und auf welcher Grundlage?

Gerade in kleineren Teams kann dies pragmatisch über Ticketsystem und Versionskontrolle erfolgen. Prüfungsrelevant ist, dass die Änderungskette für Dritte nachvollziehbar ist.

3.4.2 Versionsführung

Die Versionsführung ist kein Selbstzweck. Sie ist die Grundlage dafür, dass ein Prüfer (und später auch Anwender) **genau** verstehen kann, welcher Stand geprüft wurde.

Wesentliche Anforderungen sind:

- eindeutige Versionierung freigegebener Stände (Release-/Build-Identifikation),
- nachvollziehbare Versionshistorie,
- klare Trennung bzw. klare Regeln für Entwicklungs-, Test- und Produktivstände,
- Release Notes oder vergleichbare Änderungsübersichten.

In der Prüfungspraxis ist besonders kritisch, wenn Testnachweise nicht eindeutig einem Versionsstand zugeordnet werden können.

3.4.3 Release-Management

Release-Management stellt sicher, dass eine Softwareversion erst dann ausgeliefert wird, wenn die erforderlichen Tests abgeschlossen und die Freigabe erteilt wurde.

Prüfungsrelevant ist vor allem, dass ein Release nachvollziehbar dokumentiert:

- **welcher Stand ausgeliefert wurde**,
- **welche Tests hierfür abgeschlossen sind**,
- **welche Abweichungen ggf. bekannt sind und wie damit umgegangen wird**,
- **wer die Freigabe erteilt hat**.

Eine kurze Release-Checkliste oder ein dokumentierter Freigabevermerk genügt häufig.

3.4.4 Configuration-Management

Configuration-Management sorgt dafür, dass Konfigurationen und Umgebungsparameter nachvollziehbar sind und Test- sowie Prüfergebnisse reproduzierbar bleiben.

Für die Prüfung ist besonders relevant:

- welche Konfigurationen (Parameter, Schnittstellen, Systemvariablen) für den geprüften Stand maßgeblich sind,
- wie Konfigurationen versioniert und geändert werden (z. B. Konfigurationsdateien, IaC),
- ob Entwicklungs-, Test- und Produktivumgebung ausreichend vergleichbar sind,
- ob ein reproduzierbares Testsystem vorliegt (insbesondere für die Nachverfolgbarkeit von Testfällen).

Gerade bei konfigurierbaren Systemen kann die Konfiguration selbst prüfungsrelevant sein, weil sie die Funktionsweise des Produkts wesentlich bestimmt.

3.5 Umgang mit externen Komponenten

Moderne Softwareprodukte basieren selten ausschließlich auf eigenentwickeltem Code. Frameworks, Bibliotheken, externe Services, Cloud-Komponenten und Open-Source-Module sind häufig integraler Bestandteil des Produkts. Diese Abhängigkeiten erhöhen Geschwindigkeit und Funktionsumfang – sie erhöhen aber auch Risiken für Stabilität, Sicherheit und Reproduzierbarkeit.

Für die Prüfbarkeit ist weniger entscheidend, jede externe Komponente technisch vollständig zu kontrollieren. Entscheidend ist, dass der Hersteller den Einsatz externer Komponenten **transparent macht**, **Risiken bewertet** und **Änderungen kontrolliert**.

3.5.1 Identifikation externer Komponenten

Grundlage ist Transparenz. Der Hersteller sollte nachvollziehbar dokumentieren:

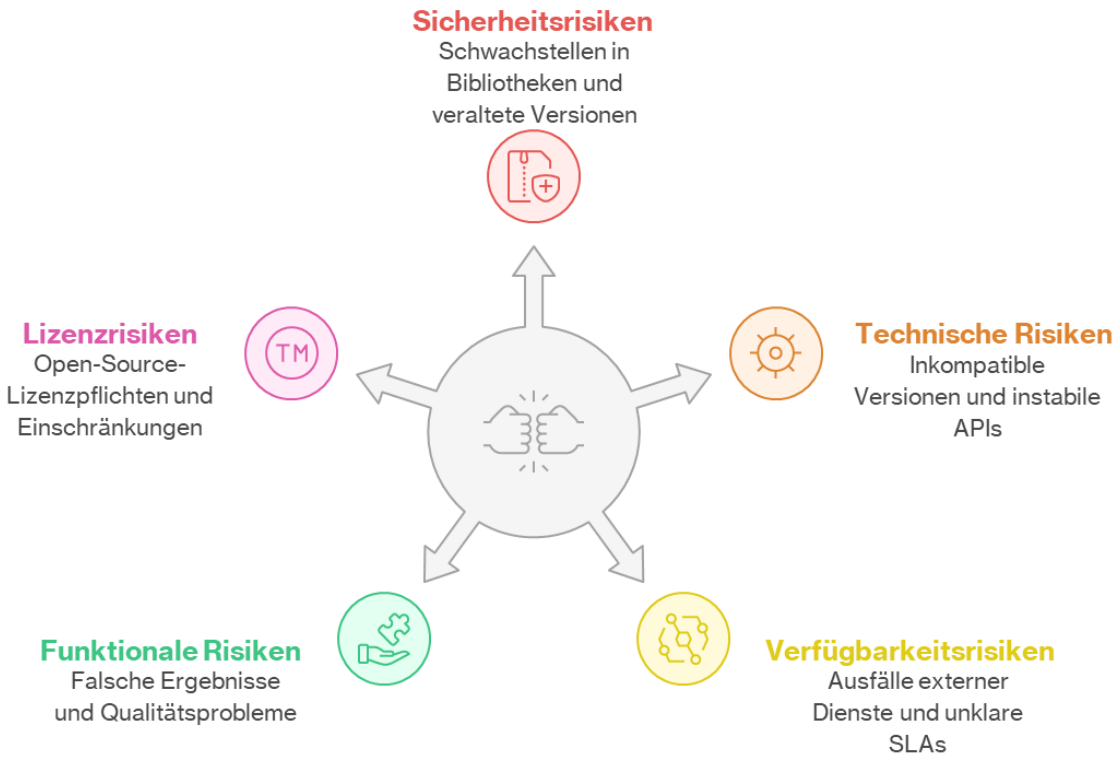
- welche externen Bibliotheken/Frameworks eingesetzt werden,
- welche Drittanbieter-Komponenten genutzt werden,
- welche externen APIs/Services angebunden sind,
- welche Cloud- oder Plattformdienste produktrelevant sind,
- welche Versionen bzw. Abhängigkeitsstände maßgeblich sind.

Diese Übersicht ist die Basis, um Verantwortung, Abhängigkeiten und Risikotreiber zu verstehen.

3.5.2 Bewertung der Risiken externer Komponenten

Externe Komponenten bringen eigenständige Risikodimensionen mit sich, die sich unmittelbar auf die Funktionsfähigkeit, Sicherheit und Stabilität des Softwareprodukts auswirken können.

Die wesentlichen Risikokategorien lassen sich wie folgt systematisieren:



Sicherheitsrisiken

Schwachstellen in Bibliotheken oder veraltete Versionen können zu Sicherheitslücken führen, die das gesamte Produkt kompromittieren. Besonders kritisch sind öffentlich bekannte CVEs, für die keine zeitnahe Bewertung oder Aktualisierung erfolgt.

Technische Risiken

Inkompatible Versionen, instabile APIs oder nicht dokumentierte Änderungen externer Dienste können zu unerwartetem Fehlverhalten führen. Gerade bei automatischen Updates cloudbasierter Services entstehen hier häufig unterschätzte Risiken.

Funktionale Risiken

Fehler oder Änderungen in externen Komponenten können unmittelbar zu falschen Verarbeitungsergebnissen führen, etwa bei OCR-Engines, Validierungsbibliotheken oder Berechnungsmodulen. Diese Risiken betreffen unmittelbar die Ordnungsmäßigkeit des Produkts.

Verfügbarkeitsrisiken

Abhängigkeiten von externen APIs oder Cloud-Diensten können zu Ausfällen führen. Unklare SLAs oder fehlende Fallback-Strategien erhöhen dieses Risiko zusätzlich.

Lizenzrisiken

Open-Source-Komponenten unterliegen Lizenzbedingungen, die Nutzungs-, Offenlegungs- oder Weitergabepflichten enthalten können. Verstöße können rechtliche oder wirtschaftliche Auswirkungen haben.

Prüferperspektive

Für die Prüfung ist nicht entscheidend, dass jedes Risiko technisch eliminiert wird. Entscheidend ist, dass:

- die Risiken identifiziert,
- nachvollziehbar bewertet,
- und mit angemessenen Maßnahmen adressiert werden.

Eine einfache, dokumentierte Risikoeinschätzung (z. B. nach Eintrittswahrscheinlichkeit und Auswirkung) ist regelmäßig ausreichend, sofern sie konsequent angewendet wird.

3.5.3 Versionierung und Aktualisierung externer Komponenten

Für die Prüfbarkeit ist entscheidend, dass externe Komponenten versionsbezogen steuerbar sind:

- dokumentierte Versionen/Bundles/Locks,
- geregelter Update-Prozess (inkl. Bewertung und Test),
- nachvollziehbare Dokumentation der Änderung (Ticket/Commit/Release Notes),
- Bewertung potenzieller Auswirkungen auf Funktionen oder Schnittstellen.

Besonders herausfordernd sind Komponenten, die sich ohne Eingreifen des Herstellers ändern (z. B. bestimmte Cloud-/API-Dienste). Hier ist Transparenz über Abhängigkeiten und Änderungen besonders wichtig.

3.5.4 Testen externer Komponenten

Externe Komponenten müssen in die Teststrategie eingebettet sein. Für die Prüfung ist darzulegen, dass:

- Schnittstellen (Integrationstests) getestet werden,
- relevante Geschäftsprozesse End-to-End funktionieren,
- Updates externer Komponenten angemessene Regressionstests auslösen,
- Testnachweise die Einbindung der externen Komponenten erkennbar abbilden.

Entscheidend ist nicht die maximale Testtiefe, sondern die nachvollziehbare Einbettung in eine konsistente Test- und Release-Logik.

4. Die Prüfung nach IDW PS 880 in der Praxis

Ein Prüfungsbericht ist kein Pflichtdokument – er ist eine Visitenkarte für Zuverlässigkeit.

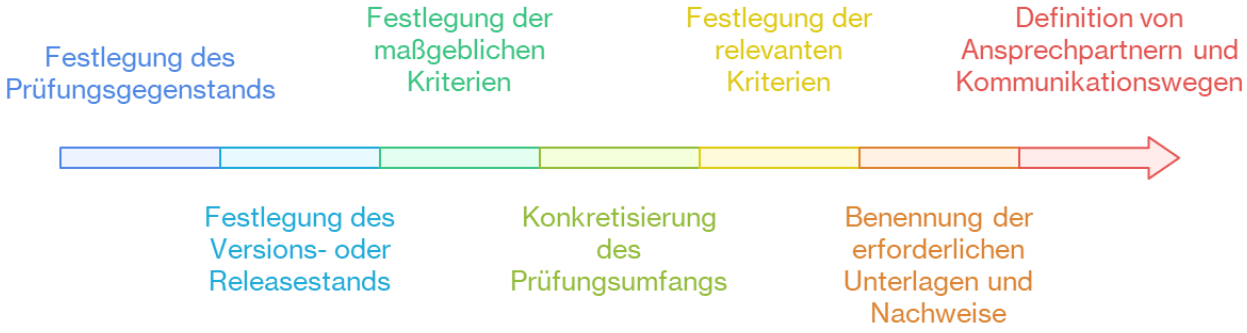
Eine Softwareprüfung nach IDW PS 880 n.F. ist kein rein formaler Vorgang. Sie ist ein strukturierter, risikoorientierter Prozess, der sowohl technisches Verständnis als auch fachliche Einordnung erfordert.

In der Praxis hängt der Prüfungsaufwand davon ab, wie strukturiert Entwicklungs-, Test- und Änderungsprozesse ausgestaltet sind.

4.1 Ablauf einer Softwareprüfung

4.1.1 Auftragsdefinition und Abgrenzung

Eine strukturierte Softwareprüfung beginnt nicht mit Testhandlungen, sondern mit klaren Definitionen. Bereits in der Auftragsphase müssen folgende Punkte eindeutig geklärt werden:



Die Qualität dieser Vorfestlegungen bestimmt maßgeblich:

- den Prüfungsumfang,
- die Tiefe der Prüfungshandlungen,
- den zeitlichen Aufwand,
- und die Erwartungshaltung aller Beteiligten.

In der Praxis entstehen viele Reibungsverluste nicht im Prüfungsprozess selbst, sondern durch unklare Abgrenzung des Prüfungsgegenstands oder uneindeutige Kriterien.

Eine saubere Definition zu Beginn wirkt daher unmittelbar auf Effizienz und Planungssicherheit.

4.1.2 Informationsgewinnung und Systemverständnis

Im diesem Schritt verschafft sich der Prüfer einen Überblick über die Software und deren Funktionsweise. Hierzu werden insbesondere die fachlichen und technischen Beschreibungen, die dokumentierten Anforderungen und Designunterlagen gesichtet.

Zentraler Bezugspunkt dieser Phase ist die **Verfahrensdokumentation** der Software. Diese beschreibt in strukturierter Form, **wie die Software fachlich konzipiert ist, wie sie funktioniert und wie sie die vorgesehenen Aufgaben erfüllt**. Sie bildet damit die Grundlage für das Verständnis der Software aus fachlicher und technischer Sicht.

Dabei kommt der **Anwenderdokumentation**, als Teil der Verfahrensdokumentation, eine besondere Bedeutung zu. Sie beschreibt die Software aus Sicht der Nutzer und zeigt, **wie die Software im vorgesehenen Nutzungskontext eingesetzt werden soll**, welche Eingaben möglich sind, welche Verarbeitungsschritte erfolgen und welche Ergebnisse erzeugt werden. Gerade bei funktionalen Prüfungen liefert die Anwenderdokumentation häufig einen unmittelbaren Zugang zu den tatsächlichen Abläufen und Geschäftsprozessen, die mit der Software abgebildet werden.

Die Verfahrens- und Anwenderdokumentation kann aus mehreren Unterlagen, u.a. fachlichen Beschreibungen, Architekturübersichten und Benutzerhandbücher, bestehen.

Entscheidend ist, dass diese Unterlagen **in ihrer Gesamtheit ein konsistentes und widerspruchsfreies Bild der Software vermitteln**.

Ziel dieser Phase ist es, zu beurteilen, ob:

- die beschriebene Funktionsweise der Software schlüssig ist,
- die Anforderungen nachvollziehbar definiert und verständlich dokumentiert sind,
- und eine geeignete Grundlage für die nachfolgenden Prüfungshandlungen, insbesondere für Tests und Nachweisprüfungen, vorliegt.

Bereits in dieser Phase identifiziert der Prüfer regelmäßig **Schwerpunkte und risikobehaftete Funktionsbereiche**, die im weiteren Prüfungsverlauf vertieft betrachtet werden. Abweichungen zwischen Verfahrens- und Anwenderdokumentation oder Unklarheiten in den Beschreibungen sind dabei häufig ein Indikator für erhöhte Prüfungsrisiken.

4.1.3 Prüfung von Design, Implementierung und Tests

In dieser Phase beurteilt der Prüfer die Ausgestaltung des Softwareentwicklungsprozesses und prüft, ob das angewandte Entwicklungs- und Testverfahren **geeignet ist, die Ordnungsmäßigkeit der Software systematisch sicherzustellen**.

Im Mittelpunkt steht dabei nicht die einzelne Softwarefunktion, sondern die Frage, **ob der Entwicklungsprozess als solcher geeignet ist**, Risiken zu identifizieren, Fehler zu vermeiden und Abweichungen rechtzeitig aufzudecken. Diese Beurteilung ist maßgeblich für die Risikoeinschätzung und bestimmt Art und Umfang der weiteren Prüfungshandlungen.

Hierzu verschafft sich der Prüfer insbesondere einen Überblick über:

- die Struktur und Nachvollziehbarkeit des Entwicklungsprozesses,
- die Anwendung geeigneter Entwicklungs- und Designstandards,
- sowie die Konzeption und Durchführung der Test- und Qualitätssicherungsmaßnahmen.

Ein besonderer Fokus liegt dabei auf den **durchgeführten Tests**. Der Prüfer beurteilt, ob Testarten, Teststufen und Testabdeckung grundsätzlich geeignet sind, die maßgeblichen Risiken der Software abzudecken und die Einhaltung der festgelegten Kriterien **nachvollziehbar zu belegen**. Dabei werden insbesondere die Testkonzepte, die Dokumentation der Testergebnisse sowie der Umgang mit Abweichungen und Freigaben gewürdigt.

Die Prüfung des Entwicklungs- und Testverfahrens erfolgt typischerweise **risikoorientiert**, **stichprobenweise** und auf Basis ausgewählter Entwicklungs-, Test- und Freigabenachweise.

Abhängig von den Ergebnissen dieser Beurteilung legt der Prüfer fest, **in welchem Umfang ergänzende eigene Prüfungshandlungen erforderlich sind**. Ein nachvollziehbar strukturierter Entwicklungs- und Testprozess mit belastbaren Nachweisen kann dazu führen, dass der Umfang zusätzlicher Funktionstests reduziert wird. Umgekehrt machen unzureichende oder lückenhafte Testnachweise regelmäßig vertiefte Prüfungen einzelner Funktionen oder Prozesse erforderlich oder können dazu führen, **dass ein Prüfungshemmnis vorliegt**.

4.1.4 Eigene Funktionstests

Auf Basis der vorangegangenen Beurteilung des Entwicklungs- und Testverfahrens entscheidet der Prüfer, ob und in welchem Umfang **eigene Funktionstests erforderlich sind**, um ein hinreichend sicheres Prüfungsurteil abgeben zu können.

Eigene Funktionstests dienen dazu, die durch den Softwarehersteller bereitgestellten Nachweise **gezielt zu ergänzen oder zu verifizieren**. Sie sind insbesondere dann erforderlich, wenn die vorhandene Test- und Nachweisdokumentation nicht ausreicht, um prüfungsrelevante Risiken abschließend zu beurteilen, oder wenn einzelne Funktionen eine besondere Bedeutung für die Einhaltung der festgelegten Kriterien haben.

Der Umfang eigener Funktionstests richtet sich dabei nicht nach dem Gesamtumfang der Software, sondern nach der **Risikoeinschätzung**. Ziel ist es nicht, sämtliche Funktionen vollständig nachzutesten, sondern ausgewählte Funktionen, Prozesse oder Verarbeitungslogiken zu prüfen, die aus Prüfungssicht wesentlich sind. Der Umfang der notwendigen eigenen Funktionstests richtet sich nach:

- der Qualität und Nachvollziehbarkeit der vorliegenden Test- und Nachweisdokumentation,
- der Komplexität und Kritikalität einzelner Softwarefunktionen,
- der Bedeutung der Software für rechnungslegungs- oder kontrollrelevante Prozesse,
- sowie den im bisherigen Prüfungsverlauf identifizierten Risiken.

Eigene Funktionstests erfolgen regelmäßig in Form **funktionaler Tests** („Blackbox-Prüfungen“). Dabei prüft der Prüfer anhand geeigneter Testfälle, ob die Software bei definierten Eingaben die erwarteten Ergebnisse erzeugt und sich im vorgesehenen Nutzungskontext ordnungsgemäß verhält. Typische Prüfungshandlungen sind der Nachvollzug zentraler Geschäftsprozesse, die Überprüfung kritischer Berechnungs- oder Entscheidungslogiken sowie die Prüfung von Validierungs- und Plausibilitätsmechanismen.

Eigene Funktionstests sind stets **ergänzender Bestandteil** der Softwareprüfung. Sie ersetzen nicht die Beurteilung des Entwicklungs- und Testverfahrens, sondern bauen darauf auf.

4.1.5 Berichterstellung und Abstimmung

Nach Abschluss der Prüfungshandlungen erstellt der Prüfer eine Entwurfsfassung des Prüfungsberichts. Diese dient der fachlichen Abstimmung und gibt dem Softwarehersteller Gelegenheit, Verständnisfragen zu klären oder sachliche Ergänzungen vorzunehmen.

Ziel der Abstimmung ist ausschließlich die Sicherstellung, dass der Prüfungsbericht sachlich korrekt ist und die getroffenen Feststellungen zutreffend und vollständig wiedergibt.

Nach Abschluss der Abstimmung wird der Prüfungsbericht finalisiert und erteilt. Er dokumentiert den Prüfungsgegenstand, die zugrunde gelegten Kriterien, den Umfang der durchgeführten Prüfung sowie das abschließende Prüfungsurteil. Der Prüfungsbericht bildet die Grundlage für die weitere Verwendung durch den Softwarehersteller, Anwenderunternehmen und deren Prüfer.

4.2 Erklärungen des Softwareherstellers

Im Rahmen der Softwareprüfung nach IDW PS 880 n.F. wird regelmäßig eine **Vollständigkeitserklärung des Softwareherstellers** eingeholt. Diese ist fester Bestandteil des Prüfungsprozesses und dient der Absicherung der Prüfungshandlungen.

Mit der Vollständigkeitserklärung bestätigen die gesetzlichen Vertreter insbesondere:

- den eindeutig abgegrenzten Prüfungsgegenstand,
- die Benennung und Vollständigkeit der maßgeblichen Kriterien,
- die vollständige Bereitstellung der für die Prüfung relevanten Verfahrens-, Test- und Abnahmedokumentationen,
- die Offenlegung eingesetzter Programme, Module, Schnittstellen sowie externer Komponenten,
- die Bereitstellung einer geeigneten Testumgebung,
- sowie die Mitteilung aller bekannten Fehler, Beeinträchtigungen, wesentlichen Mängel oder sonstigen Sachverhalte, die die Beurteilung der Software beeinflussen könnten.

Darüber hinaus umfasst die Erklärung regelmäßig Aussagen dazu, ob während der Prüfung Änderungen an der Software, am Entwicklungs- oder Testverfahren vorgenommen wurden und ob Ergebnisse aus früheren Prüfungen vorliegen.

Die Vollständigkeitserklärung ersetzt keine Prüfungshandlungen und stellt keine eigenständige Aussage über die Ordnungsmäßigkeit oder Funktionsfähigkeit der Software dar. Sie bildet jedoch eine **wesentliche Grundlage für die Durchführung der Prüfung und das abschließende Prüfungsurteil**, da sie die Verantwortung für die bereitgestellten Informationen eindeutig beim Softwarehersteller verortet.

4.3 Interpretation von Feststellungen

Im Rahmen der Softwareprüfung nach IDW PS 880 n.F. werden Feststellungen als Ergebnis der durchgeführten Prüfungshandlungen getroffen. Feststellungen können dabei sowohl **positiv** als auch **negativ** sein. Positive Feststellungen bestätigen die Ordnungsmäßigkeit einzelner Aspekte der Software oder des Softwareentwicklungsverfahrens und fließen in die Gesamtwürdigung ein. **Negative Feststellungen werden als Mängel bezeichnet.**

Die Feststellungen werden im Prüfungsbericht dokumentiert und dort **strukturiert gewürdigt**. Maßgeblich ist dabei nicht jede einzelne Feststellung für sich, sondern in einer Gesamtbetrachtung deren **Art, Gewicht und Auswirkung auf die Ordnungsmäßigkeit und Funktionsfähigkeit des Softwareprodukts**.

4.3.1 Kategorisierung von Mängeln

Negative Feststellungen (Mängel) lassen sich in der Prüfungspraxis typischerweise den folgenden Bereichen zuordnen:

- Mängel im programminternen Kontrollsystem (z. B. unzureichende Eingabe-, Verarbeitungs- oder Ausgabekontrollen),
- Mängel in Verarbeitungsfunktionen (z. B. fehlerhafte Rechenlogiken oder Periodenzuordnungen),
- Dokumentationsmängel (z. B. unvollständige oder nicht aktuelle Anwender- oder Verfahrensdokumentationen),
- Mängel in der Bedienbarkeit des Softwareprodukts.

Diese Einordnung dient der Strukturierung der Prüfungsergebnisse und unterstützt die spätere Gesamtwürdigung im Prüfungsbericht.

4.3.2 Gewichtung und Würdigung im Prüfungsbericht

Für die Ableitung der Softwarebescheinigung ist entscheidend, **wie die festgestellten Mängel im Prüfungsbericht gewichtet werden**. Dabei ist zu beurteilen, ob Mängel:

- die Ordnungsmäßigkeit und Sicherheit des Softwareprodukts **nicht wesentlich beeinträchtigen** (untergeordnete Mängel), oder
- die Ordnungsmäßigkeit, Sicherheit oder grundsätzliche Eignung des Softwareprodukts **in Frage stellen** (wesentliche Mängel).

Untergeordnete Mängel betreffen regelmäßig einzelne Detailaspekte, etwa vereinzelte fehlende Eingabekontrollen, kleinere Unstimmigkeiten in der Dokumentation oder Einschränkungen in der Bedienbarkeit. Sie haben für sich genommen **keinen Einfluss auf die Art der Softwarebescheinigung** und werden teilweise nicht gesondert im Prüfungsbericht dargestellt, sondern dem Softwarehersteller außerhalb des Berichts zur Kenntnis gegeben.

Wesentliche Mängel liegen insbesondere dann vor, wenn zentrale Verarbeitungsfunktionen fehlerhaft umgesetzt sind, rechnungslegungs- oder kontrollrelevante Anforderungen nicht ordnungsgemäß abgebildet werden oder wesentliche Anwender- oder Verfahrensdokumentationen fehlen bzw. stark lückenhaft sind. Kennzeichnend für solche Mängel ist, dass sie **regelmäßig nicht durch den Anwender kompensiert werden können**.

4.3.3 Ableitung der Softwarebescheinigung

In der Zusammenfassung der Prüfungsergebnisse werden die maßgeblichen Feststellungen, deren Würdigung sowie die daraus gezogenen Schlussfolgerungen dargestellt. Diese Zusammenfassung bildet die Grundlage für die Entscheidung über die Erteilung der Softwarebescheinigung.

Aus der Gesamtwürdigung der Prüfungsergebnisse wird abgeleitet, ob die Softwarebescheinigung uneingeschränkt erteilt, eingeschränkt erteilt oder zu versagen ist. Maßgeblich ist dabei, ob und in welchem Umfang festgestellte Mängel die Ordnungsmäßigkeit, Sicherheit oder Eignung des Softwareprodukts beeinträchtigen.

Die Softwarebescheinigung enthält die abschließende Aussage darüber, ob das geprüfte Softwareprodukt in dem geprüften Versions- oder Releasestand bei sachgerechter Anwendung den im Prüfungsauftrag vereinbarten Kriterien entspricht.

5. Erfolgsfaktoren und typische Stolpersteine

Die meisten Prüfungshemmnisse entstehen nicht im Code – sondern in fehlender Struktur.

Die vorstehenden Kapitel zeigen, welche fachlichen, organisatorischen und dokumentarischen Anforderungen an eine prüfbare Softwareentwicklung nach IDW PS 880 n.F. zu stellen sind. In der Praxis entscheidet jedoch weniger die formale Ausgestaltung einzelner Prozesse, sondern deren **konsistente Anwendung und Nachvollziehbarkeit** über den Erfolg einer Softwareprüfung.

Die nachfolgenden Erfolgsfaktoren und typischen Stolpersteine fassen zentrale Erfahrungen aus der Prüfungspraxis zusammen.

5.1 Erfolgsfaktoren

Die Erfahrung aus Softwareprüfungen zeigt, dass folgende Faktoren maßgeblich zum Gelingen beitragen:

- **Klare Abgrenzung des Prüfungsgegenstands**

  Eine frühzeitige und eindeutige Festlegung des geprüften Softwareprodukts bzw. des konkreten Versions- oder Releasestandes verhindert Missverständnisse und spätere Prüfungshemmnisse.

- **Geeignete und nachvollziehbar definierte Kriterien**

  Erfolgreiche Prüfungen basieren auf klar dokumentierten Kriterien, die für das Aufgabengebiet der Software relevant, vollständig und verständlich sind. Unklare oder implizite Kriterien führen regelmäßig zu Diskussionen im Prüfungsverlauf.

- **Nachvollziehbarer Entwicklungs- und Testprozess**

  Nicht die Komplexität, sondern die Konsistenz des Vorgehens ist entscheidend. Ein strukturierter Entwicklungsprozess mit klaren Übergängen zwischen Anforderungen, Design, Implementierung, Tests und Freigabe bildet die Basis für die Prüfbarkeit.

- **Belastbare Test- und Nachweisdokumentation**

  Tests sind der zentrale Nachweis für die Ordnungsmäßigkeit der Software. Erfolgreich sind insbesondere solche Prüfungen, bei denen Testkonzepte, Testergebnisse und Freigaben verständlich dokumentiert und eindeutig nachvollziehbar sind.

- **Pragmatischer Umgang mit Formalitäten**

  Prüfungsfähigkeit erfordert keine übermäßige Bürokratie. Schlanke, aber konsistente Dokumentationen und Prozesse sind gerade in kleineren Entwicklerteams häufig wirksamer als umfangreiche, aber nicht gelebte Regelwerke.

- **Frühzeitige Einbindung des Prüfers**

  Eine frühzeitige Abstimmung zu Prüfungsgegenstand, Kriterien und Nachweisen reduziert den Prüfungsaufwand und vermeidet spätere Überraschungen

5.2 Typische Stolpersteine

Demgegenüber zeigen sich in der Prüfungspraxis regelmäßig wiederkehrende Problembereiche:

- **Unklare oder wechselnde Prüfungsabgrenzung**

  Änderungen am geprüften Versionsstand oder am Funktionsumfang während der Prüfung führen häufig zu Prüfungshemmnissen oder Einschränkungen der Bescheinigung.

- **Unzureichend dokumentierte Anforderungen**

  Wenn fachliche Anforderungen nur implizit bekannt oder nicht ausreichend dokumentiert sind, fehlt die Grundlage für eine belastbare Beurteilung von Design, Implementierung und Tests.

- **Tests ohne nachvollziehbare Nachweise**

  Nicht dokumentierte Tests gelten aus Prüfersicht als nicht durchgeführt. Fehlende oder unvollständige Testergebnisse sind eine der häufigsten Ursachen für Feststellungen.

- **Vermischung von Entwicklung und Betrieb**

  Unklare Trennung zwischen Entwicklungs-, Test- und Produktivumgebungen erschwert die Nachvollziehbarkeit und führt zu Unsicherheiten hinsichtlich des geprüften Zustands.

- **Unterschätzung externer Komponenten**

  Fehlende Transparenz über eingesetzte Bibliotheken, Frameworks oder externe Services kann zu wesentlichen Feststellungen führen, insbesondere im Hinblick auf Sicherheit und Stabilität.

- **Falsche Erwartungshaltung an die Softwarebescheinigung**

  Die Annahme, eine Softwarebescheinigung ersetze interne Kontrollen, Anwendersorgfalt oder eine sachgerechte Parametrisierung, führt regelmäßig zu Fehlinterpretationen des Prüfungsumfangs

Hinweis:

Das Whitepaper wurde von den Geschäftsführern der R3 Audit & Assurance GmbH Wirtschaftsprüfungsgesellschaft erstellt. Es dient der allgemeinen Information und ersetzt keine individuelle rechtliche oder prüferische Beratung.

Die R3 Audit & Assurance GmbH Wirtschaftsprüfungsgesellschaft unterstützt Softwarehersteller und Anwenderunternehmen bei der Einordnung, Vorbereitung und Durchführung von Softwareprüfungen nach IDW PS 880 n.F. Dies umfasst insbesondere die Beurteilung der Prüfbarkeit von Softwareentwicklungsprozessen, die Begleitung bei der Vorbereitung auf Softwareprüfungen sowie die unabhängige Prüfung von Softwareprodukten.

www.r3audit.de