Hochschule Neu-Ulm
University of Applied Sciences

Master-
arbeit im
Studiengang
Strategic Information Management
an der Hochschule für angewandte
Wissenschaften Neu-Ulm

Vergleich von traditionellen und KI-basierten Testverfahren für Dokumentenmanagementsysteme – Entwicklung einer Entscheidungsunterstützung mittels MCDA und DSR

Erstkorrektor	Prof. Oliver Grieble
Zweitkorrektor:	Prof. Joerg-Oliver Vogt

Autor:	Christian Koch (351760)

Thema erhalten:	07.04.2025
Arbeit abgegeben:	04.08.2025

Hochschule Neu-Ulm
University of Applied Sciences

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die zum Gelingen dieser Masterarbeit beigetragen haben.

Mein besonderer Dank gilt Prof. Oliver Grieble, der mir als Erstkorrektor zur Seite stand und mich im Laufe der Arbeit immer wieder mit wertvollen Hinweisen und Anregungen unterstützt hat. Ebenso danke ich Prof. Joerg-Oliver Vogt für die Übernahme der Zweitkorrektur.

Ein herzliches Dankeschön geht auch an meinen Praxispartner aktivweb System- und Datentechnik GmbH. Besonders bedanken möchte ich mich bei Uwe Weikl, der mir die Möglichkeit gegeben hat, die Masterarbeit im Unternehmen zu schreiben, sowie bei Philip Dachs, der mich bei technischen Fragestellungen und in der Umsetzung tatkräftig unterstützt hat.

Vielen Dank für das Vertrauen und die gute Zusammenarbeit!

Hochschule Neu-Ulm
University of Applied Sciences

Erklärung zum Einsatz Künstlicher Intelligenz

Ich erkläre hiermit, dass ich im Rahmen der Erstellung dieser Masterarbeit Werkzeuge der Künstlichen Intelligenz unterstützend eingesetzt habe. Konkret kamen dabei folgende Tools zum Einsatz:

- **ChatGPT** wurde genutzt, um Stichpunkte in zusammenhängenden wissenschaftlichen Text zu überführen, grammatikalische und stilistische Optimierungen vorzunehmen sowie beim Brainstorming von Ideen und Argumentationsstrukturen zu unterstützen.
- **DeepL Write** diente zur sprachlichen Verfeinerung und zur Sicherstellung eines konsistenten, akademisch angemessenen Schreibstils.
- **Elicit** wurde ergänzend zur Literaturrecherche eingesetzt, insbesondere zur Identifikation relevanter wissenschaftlicher Quellen.

Alle durch KI unterstützten Inhalte wurden von mir sorgfältig geprüft, überarbeitet und eigenständig in den Gesamtkontext der Arbeit integriert. Die inhaltliche Verantwortung für Struktur, Argumentation und wissenschaftliche Aussagekraft der Arbeit liegt vollständig bei mir als Autor dieser Masterarbeit.

Hochschule Neu-Ulm
University of Applied Sciences

Inhaltsverzeichnis

Hochschule Neu-Ulm
University of Applied Sciences

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Abkürzung	
API	Application Programming Interface
CD	Continuous Deployment
CI	Continuous Integration
CRM	Customer Relationship Management
DMS	Dokumentenmanagementsystem
DOM	Document Object Model
DSGVO	Datenschutz-Grundverordnung
DSR	Design Science Research
ERP	Enterprise Resource Planning
E2E	End-to-End (Testing)

Abkürzung

GoBD	Grundsätze zur ordnungsmäßigen Führung und Aufbewahrung von Büchern, Aufzeichnungen und Unterlagen in elektronischer Form
GUI	Graphical User Interface
KMU	Kleine und mittlere Unternehmen
MCDA	Multi-Criteria Decision Analysis
OCR	Optical Character Recognition
RPA	Robotic Process Automation
UAT	User Acceptance Testing

1. Einleitung

1.1 Motivation und Relevanz des Themas

In den letzten Jahren hat sich die Künstliche Intelligenz rasant weiterentwickelt. Sie kommt heute längst nicht mehr nur in klassischen Anwendungsfeldern, wie der maschinellen Übersetzung oder der thematische Einordnung von Texten zum Einsatz, sondern zunehmend auch in komplexeren Feldern. Einer dieser neuen Anwendungsbereiche ist die Softwareentwicklung. Hier wird Künstliche Intelligenz nicht nur zur automatisierten Codegenerierung, sondern auch im Software-Testing bis hin zu vollständig automatisierten Testabläufen genutzt (Ahmed Ramadan, 2024).

KI-basierte Testverfahren kombinieren maschinelles Lernen mit automatisierten Testprozessen und versprechen dadurch eine höhere Effizienz sowie eine Reduzierung des manuellen Aufwands (Martensson, 2022). Besonders relevant ist dieser Fortschritt für das sogenannte End-to-End-Testing. Bei dieser Testmethode werden vollständige Anwendungsszenarien durchlaufen, die die Interaktion zwischen System und Nutzer möglichst realitätsnah simulieren. Dabei wird nicht nur eine einzelne Komponente isoliert geprüft. Vielmehr wird die gesamte Software getestet, von der Benutzeroberfläche (GUI) über Backend Prozesse bis hin zur Einbindung der Datenbank. Ziel ist es, die Funktionalität und Stabilität der gesamten Anwendung sicherzustellen (ISTQB, 2025).

Aktuelle Studien zeigen, dass der Einsatz von KI branchenübergreifend stark zunimmt, denn 72 % der Unternehmen weltweit nutzen bereits KI (Demetrio, 2024), und die Tendenz ist steigend. Gleichzeitig nutzen nur 9 % der kleinen und mittleren Unternehmen (KMU) in Deutschland überhaupt KI-Technologien (Rammer, 2024). Dieses Ungleichgewicht zeigt eine zentrale Herausforderung, denn während große Unternehmen von KI-gestützten Innovationen profitieren, fehlen KMU häufig die nötigen Strukturen, Ressourcen und Entscheidungsgrundlagen.

Gerade im Bereich des Software-Testings, der durch hohe Komplexität, Fachkräftemangel und steigende Qualitätsanforderungen geprägt ist, stellt sich für KMU die Frage, welche Teststrategie den größten Nutzen bietet, ohne die Unternehmen finanziell oder organisatorisch zu überfordern.

Aus diesem Grund gewinnt die Frage nach der geeigneten End-to-End-Testmethode für kleine und mittlere Unternehmen zunehmend an Relevanz. Ein direkter, systematischer Vergleich zwischen traditionellen und KI-gestützten End-to-End-Testverfahren fehlt bislang in der wissenschaftlichen Literatur.

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist es, traditionelle und KI-gestützte End-to-End-Testverfahren hinsichtlich ihrer hinsichtlich ihres Nutzens für den Einsatz in kleinen und mittleren Unternehmen (KMU) systematisch zu vergleichen. Vor dem Hintergrund begrenzter Ressourcen und steigender Anforderungen an Softwarequalität stehen KMU vor der Herausforderung, geeignete Teststrategien auszuwählen, die sowohl technisch wirksam als auch wirtschaftlich sinnvoll sind.

Im Fokus der Untersuchung stehen dabei sowohl funktionale als auch nicht-funktionale Bewertungskriterien. Die Analyse erfolgt anhand eines praxisnahen Anwendungsszenarios aus dem Bereich der Dokumentenmanagementsysteme (DMS), die sich aufgrund ihrer Komplexität und zentrale Bedeutung für Geschäftsprozesse als geeignetes Untersuchungsfeld anbieten.

Zur strukturierten Bewertung wird eine Multi-Criteria Decision Analysis (MCDA) herangezogen. Dieses Entscheidungsmodell ermöglicht es, verschiedene Kriterien zu gewichten und auf transparente Weise vergleichbar zu machen. Die MCDA-Matrix soll als Instrument dienen, um Unternehmen eine valide Entscheidungsunterstützung bei der Auswahl eines passenden End-to-End-Testverfahrens zu bieten.

Damit verfolgt die Arbeit sowohl ein wissenschaftliches als auch ein praxisorientiertes Ziel, denn einerseits soll ein Beitrag zur systematischen Bewertung moderner Testmethoden geleistet werden, andererseits soll ein konkreter Mehrwert für Entscheidungsträger im Unternehmenskontext geschaffen werden.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in insgesamt sechs Kapitel.

Kapitel 1 liefert die Einführung in das Thema, erläutert die Relevanz von KI-gestützten Testverfahren im Softwareentwicklungsprozess und beschreibt die zugrunde liegende Problemstellung. Darüber hinaus werden die Zielsetzung und der Aufbau der Arbeit vorgestellt.

Kapitel 2 gibt einen Überblick über den theoretischen Hintergrund. Dazu zählen die Grundlagen der Künstlichen Intelligenz im Software-Testing, das Konzept des End-to-End-Testings sowie ein Vergleich klassischer und KI-gestützter Testansätze. Ebenso werden die besonderen Anforderungen und Rahmenbedingungen in kleinen und mittleren Unternehmen (KMU) sowie die spezifischen Eigenschaften von Dokumentenmanagementsystemen (DMS) näher beleuchtet.

Kapitel 3 widmet sich der methodischen Vorgehensweise. Hier wird die Auswahl und Anwendung der Multi-Criteria Decision Analysis (MCDA) als Bewertungsinstrument erläutert. Zudem werden die Bewertungskriterien sowie die zugrunde liegenden Gewichtungen und Vergleichsparameter beschrieben.

Kapitel 4 beinhaltet die praktische Anwendung der MCDA auf die beiden untersuchten Testansätze im Kontext eines DMS-Szenarios. Es erfolgt eine vergleichende Analyse der Ergebnisse, ergänzt durch eine kritische Reflexion.

Kapitel 5 diskutiert die zentralen Erkenntnisse der Untersuchung. Dabei werden die Implikationen für die Praxis, insbesondere für KMU, sowie mögliche Limitationen der Untersuchung betrachtet.

Kapitel 6 schließt die Arbeit mit einem Fazit ab und gibt einen Ausblick auf zukünftige Forschungsvorhaben im Bereich KI-gestützter Testverfahren.

1.4 Design Science Research (DSR)

Die vorliegende Arbeit basiert methodisch auf dem Design Science Research (DSR)-Modell, welches in der Wirtschaftsinformatik ein weit verbreiteter Forschungsansatz ist. Dieser verfolgt einen lösungsorientierten Ansatz, bei dem durch wissenschaftliches Arbeiten ein sogenanntes Artefakt entwickelt wird, dies bietet eine greifbare Lösung für ein konkretes und praxisnahes Problem (Peffers, 2007).

Im Rahmen dieser Arbeit wird ein Artefakt in Form eines MCDA-Bewertungsmodells (Multi-Criteria Decision Analysis) entwickelt. Ziel ist es, traditionelle und KI-gestützte End-to-End-Testverfahren systematisch zu vergleichen. Ein besonderer Fokus liegt hierbei auf den Anforderungen und Rahmenbedingungen von KMU‘s.

Die Methodik orientiert sich am Design Science Research (DSR)-Ansatz nach Peffers et al. (2007) und durchläuft die folgenden fünf iterativen Phasen:

1. Problemidentifikation:
   Die Auswahl geeigneter Testverfahren für KMU ist mit Unsicherheiten behaftet. Der Mangel an vergleichenden wissenschaftlichen Analysen bildet die Ausgangslage dieser Arbeit.
2. Zieldefinition:
   Basierend auf der Problemstellung wird das Ziel verfolgt, ein anwendungsnahes Bewertungsmodell zur Entscheidungsunterstützung zu entwickeln.
3. Entwicklung des Artefakts:
   Es wird eine MCDA-Matrix entworfen, mit der sich verschiedene Testverfahren anhand technischer und ökonomischer Kriterien systematisch bewerten lassen.
4. Demonstration:
   Die Anwendbarkeit des Modells wird exemplarisch anhand eines praxisnahen Szenarios aus dem Bereich von Dokumentenmanagementsystemen (DMS) demonstriert.

5. Evaluation:
   Das Artefakt wird kritisch reflektiert und hinsichtlich seiner Eignung und Aussagekraft im definierten Anwendungskontext überprüft.

Insgesamt eignet sich der Design-Science-Ansatz besonders für diese Arbeit, da er die Verbindung zwischen wissenschaftlicher Erkenntnisgewinnung und praktischer Relevanz ermöglicht.

2. Theoretische Grundlagen

2.1 Dokumentenmanagementsysteme (DMS)

2.1.1 Definition

Ein Dokumentenmanagementsystem (DMS) ist eine spezialisierte Softwarelösung zur strukturierten Erfassung, Verwaltung, Speicherung, Bereitstellung und Archivierung elektronischer Dokumente. Dazu zählen nicht nur die digitalen Originaldokumente, sondern auch die physische Dokumente, die durch digitales Einlesen in ein elektronisches Format gebracht wurden (Kampffmeyer & Merkel, 1999). Ein wesentliches Ziel eines DMS ist es, die Verfügbarkeit und Nachvollziehbarkeit von Informationen über den gesamten Durchlauf eines Dokuments sicherzustellen.

Laut Kampffmeyer & Merkel (1999) steuern DMS den gesamten Lebenszyklus eines Dokuments von der Erstellung bis zur rechtskonformen Archivierung. Im Unterschied zu einfachen Dateiablagen bieten Dokumentenmanagementsysteme zentrale Funktionen wie Versionierung, Metadatenmanagement und differenzierte Zugriffskontrolle. Diese Merkmale ermöglichen eine strukturierte Ablage und eine effiziente Suche von Informationen.

Ein weiterer Vorteil liegt in der zentralen Bereitstellung von Informationen. In vielen Organisationen bestehen Informationssilos, die durch private Speicherungen oder unstrukturierte Dateiablagen entstehen. Genau dagegen bieten DMS hier eine einheitliche Plattform, die redundante Speicherungen verringert und die Zusammenarbeit innerhalb und zwischen Abteilungen verbessert (*The State of Intelligent Information Management. Association for Intelligent Information Management.*, 2023).

2.1.2 Entwicklung

Die Entwicklung von DMS lässt sich in mehrere Phasen einteilen. In den 1980er-Jahren wurden erste Systeme eingeführt, die vorrangig der elektronischen Archivierung gescannter Dokumente dienten und einfache Suchfunktionen bereitstellten (O'Callaghan & Smits,

2005). In den 1990er-Jahren erweiterten sich die Funktionalitäten um Versionierung, Metadatenverwaltung und erste Workflow-Komponenten (Kampffmeyer & Merkel, 1999). Seit den 2000er-Jahren ermöglichen DMS Steuerungen der Dokumente, die sich an den Prozessen eines Unternehmen orientieren. Außerdem bieten sie gemeinsame Bearbeitung und revisionssichere Archivierung ("Market Guide for Content Services Platforms," 2022). Mit der zunehmenden Digitalisierung ist auch die Erwartungshaltung an DMS gestiegen. Die Systeme müssen heute nicht nur verwalten, sondern aktiv zur Optimierung von Geschäftsprozessen mitwirken. Dazu gehören z. B. die automatische Dokumentenklassifikation, intelligente Workflows und Integrationen zu anderen Systemen wie ERP oder CRM. Diese Entwicklungen haben DMS von einfachen Ablagesystemen zu wichtigen Tools für die Prozessgestaltung weiterentwickelt (Riggert, 2019).

2.1.3 Dokumentenlebenszyklus

Der Dokumentenlebenszyklus beschreibt die verschiedenen Phasen, die ein Dokument innerhalb eines DMS durchläuft. Laut AIIM (*The State of Intelligent Information Management. Association for Intelligent Information Management.*, 2023) beinhaltet dieser Zyklus die Erfassung, Verwaltung, Bereitstellung und Aufbewahrung.

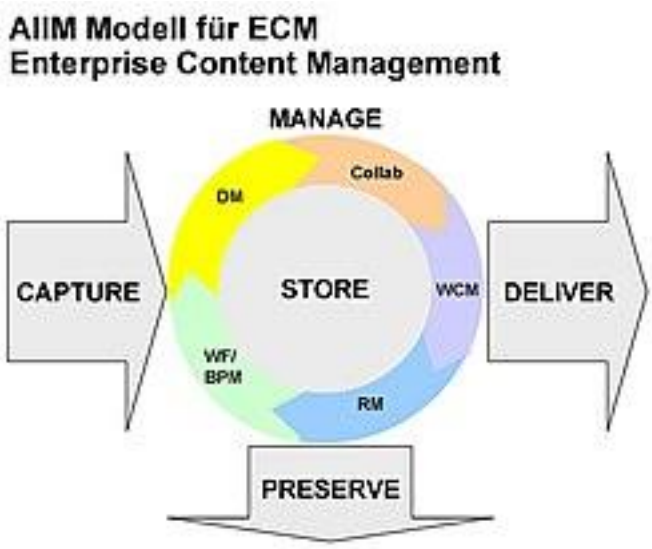

Abbildung 1: Dokumentenzyklus

Erfassungs- und Erstellungsphase

Die erste Phase im Lebenszyklus ist die Erfassung oder Erstellung von Dokumenten in digitaler Form. Digitale Dokumente können auf unterschiedliche Weise entstehen, zum Beispiel indem sie direkt erstellt, aus bestehenden IT-Systemen übernommen oder durch das Scannen von Papierdokumenten digitalisiert werden. Moderne DMS bieten hierfür verschiedene Möglichkeiten und setzen häufig auf automatische Texterkennung (OCR) sowie eine Volltextindizierung, um die spätere Suche und Wiederauffindbarkeit der Dokumente zu erleichtern (Ostheimer & Janz, 2005).

Ein praktisches Beispiel hierfür ist die DOCBOX, die es ermöglicht, Dokumente aus unterschiedlichen Quellen wie Scannern, E-Mails oder Dateiuploads automatisiert zu erfassen und direkt strukturiert abzulegen. Dadurch wird bereits bei der Erfassung eine geordnete Ablage und schnelle Auffindbarkeit gewährleistet (DocBox, 2025a).

Bearbeitungs- und Verwaltungsphase

Nach der Erfassung folgt die Bearbeitungs- und Verwaltungsphase. Hier können die Dokumente kategorisiert und versioniert werden. Außerdem ist es möglich, die Dokumente mit Metadaten und Zugriffsrechten zu versehen. Die strukturierte Ablage und Schlagwortzuordnung ermöglichen eine effiziente Organisation und schnelle Suche. Die Versionierung sorgt dafür, dass Änderungen nachvollziehbar bleiben und ältere Dokumentenstände bei Bedarf wiederhergestellt werden können. Die Zugriffsrechte werden so vergeben, dass die User nur die Dokumente sehen und bearbeiten können, für die sie auch berechtigt sind, was insbesondere im Hinblick auf Datenschutz und Compliance von großer Bedeutung ist (Ostheimer & Janz, 2005).

Die DOCBOX unterstützt diese Prozesse durch eine rollenbasierte Rechtevergabe und gewährleistet eine revisionssichere Speicherung, die den gesetzlichen Anforderungen entspricht (DocBox, 2025a).

Speicherung und Bereitstellung

Im Anschluss werden die Dokumente zentral gespeichert und können den Usern standortunabhängig bereitgestellt werden. DMS bieten in dieser Phase leistungsfähige Suchfunktionen, Workflow-Management und Schnittstellen zu anderen Anwendungen, wodurch automatisierte Prozesse ermöglicht werden (Kampffmeyer & Merkel, 1999).
Die DOCBOX ermöglicht beispielsweise Zugriff über einen Browser, was dazu führt, dass relevante Informationen jederzeit und überall verfügbar sind (DocBox, 2025a).

Archivierungs- und Löschphase
Die letzte Phase im Lebenszyklus eines Dokuments ist die revisionssichere Archivierung und nach Ablauf gesetzlicher Aufbewahrungsfristen die datenschutzkonforme Löschung. In dieser Phase werden alle Zugriffe und Änderungen protokolliert, sodass die Nachvollziehbarkeit und Rechtssicherheit stets gewährleistet ist (Kampffmeyer & Merkel, 1999).
Die DOCBOX erfüllt diese Anforderungen durch eine sichere, unveränderbare und somit nachvollziehbare Archivierung und unterstützt Unternehmen dabei, gesetzliche Vorgaben wie die GoBD und die DSGVO einzuhalten (DocBox, 2025b).

Durch die konsequente Anwendung dieser Phasen und Funktionen trägt ein DMS wesentlich zur Effizienzsteigerung, Transparenz und Rechtssicherheit im Unternehmen bei.

2.1.4 Technologische Entwicklungen

Während Dokumentenmanagementsysteme (DMS) ursprünglich meist lokal in den IT-Infrastrukturen der Unternehmen (On-Premise) implementiert wurden, verlagert sich der Trend zunehmend hin zu Cloud-basierten und hybriden Architekturen. Diese Entwicklung ermöglicht eine flexible Skalierbarkeit, eine standortunabhängige Nutzung und eine einfachere Integration in bestehende Unternehmenslandschaften. Moderne DMS bieten offene Schnittstellen (APIs), um eine nahtlose Anbindung an Systeme wie etwa an Enterprise-Resource-Planning-Systeme (ERP) oder ein Customer-Relationship-Management-Systeme (CRM) (Ostheimer & Janz, 2005).

Darüber hinaus gewinnt die Nutzung mobiler Endgeräte für die Dokumentenbearbeitung immer mehr an Bedeutung. Mobile DMS-Lösungen ermöglichen es Mitarbeitenden, einfach auf Dokumente zuzugreifen und diese zu bearbeiten, was die Agilität und Produktivität im Unternehmen deutlich erhöht (Ostheimer & Janz, 2005).

Ein weiteres wichtiges Technologiefeld ist die Automatisierung wiederkehrender Abläufe durch Robotic Process Automation (RPA). So können beispielsweise E-Mails, Formulare oder andere standardisierte Dokumente automatisiert archiviert und verarbeitet werden, was zu einer erheblichen Entlastung der Mitarbeitenden und einer Reduzierung von Fehlerquellen beiträgt (*The State of Intelligent Information Management. Association for Intelligent Information Management.*, 2023).

Ein neues Innovationsfeld ist der Einsatz von Künstlicher Intelligenz (KI) und maschinellem Lernen. Diese Technologien werden in DMS zunehmend zur automatischen Klassifikation von Dokumenten, zur Extraktion relevanter Inhalte sowie zur Unterstützung von Suchfunktionen eingesetzt. Dadurch lassen sich große Mengen unstrukturierter Daten verwalten und somit auch relevante Informationen schneller herausarbeiten (*The State of Intelligent Information Management. Association for Intelligent Information Management.*, 2023).

2.1.5 Rechtliche Anforderungen

Die Nutzung von DMS unterliegt strengen regulatorischen Anforderungen. Zentrale Rechtsgrundlagen sind insbesondere die Datenschutz-Grundverordnung (DSGVO), die eIDAS-Verordnung speziell in Deutschland die GoBD-Richtlinien. DMS müssen sicherstellen, dass gespeicherte Dokumente unveränderbar, nachvollziehbar und verfügbar bleiben.

Besonders kritisch ist die rechtskonforme Langzeitarchivierung: Dokumente müssen oft über Jahre hinweg in ihrer Originalform abrufbar sein. Hinzu kommen branchenspezifische Regelungen, etwa für Gesundheitswesen, Finanzdienstleister oder öffentliche Verwaltungen. Ein DMS muss daher flexibel konfigurierbar sein, um unterschiedlichen Compliance-Anforderungen gerecht zu werden.

2.1.6 Herausforderungen für Testing

Die Implementierung und der Betrieb von Testverfahren im Kontext von Dokumentenmanagementsystemen (DMS) sind auch mit spezifischen Herausforderungen verbunden. Diese ergeben sich zum einen durch technischen Anforderungen, aber auch zum Anderen aus organisatorischen und regulatorischen Anforderungen. DMS sind in der Regel tief in bestehende Unternehmensarchitekturen integriert und stehen in direkter Verbindung mit anderen internen und externen Systemen. Diese Kommunikation mit anderen Systemen bringt eine Abhängigkeit mit sich, was eine Herausforderung für das Testing ist.

Ein weiteres Problem ergibt sich aus der zunehmenden Nutzung dynamischer, webbasierter Benutzeroberflächen, welche oft clientseitige Logiken direkt im Browser ausführen, was oft ihr Verhalten schwer vorhersehbar macht, da es von unterschiedlichen Geräten, Browsern und Nutzereingaben abhängt.

Außerdem führen Änderungen an der DOM-Struktur, wie CSS-Selektoren oder unterschiedliche Ladezeiten bei automatisierten UI-Tests oft zu Instabilitäten. (Sampath & Sprenkle, 2016).

Des Weiteren stellt die Nutzung realistischer, aber auch datenschutzkonformer Testdaten eine zentrale Herausforderung dar, denn DMS verarbeiten häufig personenbezogene oder geschäftskritische Daten, die bei der Testdurchführung angemessen geschützt und anonymisiert werden müssen. In der Praxis ist die Einrichtung der Testumgebungen mit hohem Aufwand verbunden und nur begrenzt automatisierbar (Rwemalika et al., 2018).

Darüber hinaus erfordern die vielen und schnellen Änderungen in DMS-Projekten eine kontinuierliche Pflege der Testfälle. Bereits durch kleinere Änderungen an Workflows oder Metadatenstrukturen können umfangreiche Anpassungen in den Testskripten notwendig sein. Dies erhöht den Wartungsaufwand und belastet vor allem kleinere Unternehmen, die nicht über spezialisierte ausreichende Ressourcen verfügen (Garousi, 2022).

Nicht zuletzt wirken sich begrenzte personelle und technische Ressourcen in kleinen und mittleren Unternehmen (KMU) auf die Umsetzbarkeit von Testautomatisierung aus. Komplexe Tools mit hohen Einstiegshürden sind für viele KMU ungeeignet. Es besteht daher ein konkreter Bedarf an einfach einführbaren, wartungsarmen und zugleich leistungsfähigen Testlösungen.

Diese Herausforderungen verdeutlichen die besondere Komplexität der Testautomatisierung im DMS-Umfeld. Sowohl traditionelle als auch KI-gestützte Verfahren müssen unter diesen Rahmenbedingungen bewertet werden, um praxisgerechte und wirtschaftlich sinnvolle Testverfahren für Unternehmen zu ermöglichen.

2.2 Testverfahren: Traditionell vs. KI-gestützt

2.2.1 Grundlagen Softwaretesting im Web

Web Application Testing

Webanwendungen bilden heute einen zentralen Bestandteil digitaler Geschäftsprozesse. Sie haben auf unterschiedliche Faktoren einen direkten Einfluss, wie zum Beispiel auf Benutzererfahrung oder Kundenzufriedenheit, aber auch interne Faktoren wie betriebliche Effizienz. Web Application Testing bezeichnet die systematische Überprüfung von Webanwendungen hinsichtlich verschiedener Kriterien, wie beispielsweise Funktionalität, Benutzerfreundlichkeit, Sicherheit, Kompatibilität und Performance (B. Garcia F. Gortazar & Jim´enez, 2017). Ziel ist es, Fehler möglichst frühzeitig zu erkennen und zu verhindern, dass sie in den Live-Betrieb kommen und dem Kunden zur Verfügung gestellt werden.

Eine Webanwendung setzt sich in der Regel aus den drei klassischen Komponenten Frontend, Backend und Datenbank zusammen. Im Kontext eines DMS werden in der Regel zusätzlich Authentifizierungsdienste und externe Schnittstellen integriert. Diese Architektur stellt besondere Anforderungen an das Testen, da alle Komponenten sowohl einzeln als auch im Zusammenspiel korrekt funktionieren müssen. Ein besonderes Augenmerk liegt auf der Interaktion zwischen Benutzer und System, weshalb funktionale Tests, UI-Tests, Usability-Tests und Integrationstests fester Bestandteil des Web Application Testing sind. Aufgrund der Dynamik von Webanwendungen, die häufig mit JavaScript, AJAX oder Single-Page-Technologien arbeiten, müssen Tests regelmäßig aktualisiert werden, um Änderungen an der Benutzeroberfläche zu berücksichtigen (Sampath & Sprenkle, 2016).

Herausforderungen Web Application Testing

Das Testen von Webanwendungen ist mit einer Vielzahl technischer und organisatorischer Herausforderungen verbunden. Eine der signifikanten Schwierigkeiten besteht in der Diversität der Zielsysteme. Anwendungen müssen in unterschiedlichen Browsern (z. B. Chrome, Firefox, Safari) auf verschiedenen Betriebssystemen (Windows, macOS, Linux) und Endgeräten (Desktop, Tablet, Smartphone) zuverlässig funktionieren. Diese Vielfalt erfordert eine hohe Anzahl von Testfällen und erhöht den Aufwand für das Testdesign extrem. Ein weiteres Problem besteht darin, dass moderne Webanwendungen Inhalte häufig dynamisch im Hintergrund laden und viele Funktionen direkt im Browser umgesetzt werden. Dadurch kann es zu zeitlichen Abweichungen kommen, was die Reproduzierbarkeit von Tests erschwert. Hinzu kommt, dass solche Anwendungen oft von Backend-Systemen oder externen Diensten abhängig sind, die sich nur schwer isoliert testen lassen.

Zudem stellt die Gewährleistung der Datenkonsistenz eine zentrale Herausforderung dar: Webanwendungen nutzen häufig persistente Datenspeicher, was bedeutet, dass die Testdaten immer gleich und nachvollziehbar sein müssen, um verlässliche Ergebnisse zu erzielen. In der Praxis erweisen sich Testumgebungen jedoch oft als instabil, inkonsistent oder nur eingeschränkt verfügbar, was fehlerhafte Testergebnisse zur Folge haben kann. (Rwemalika et al., 2018).

Automatisiertes Testing

Testautomatisierung ist eine zentrale Strategie zur Bewältigung der genannten Herausforderungen. Automatisierte Tests bieten den Vorteil, dass sie reproduzierbar, schnell ausführbar und kontinuierlich in CI/CD-Pipelines integrierbar sind. Insbesondere Regressionstests profitieren von Automatisierung, da sie regelmäßig bei jeder Änderung des Quellcodes ausgeführt werden können.

Die Automatisierung kann auf unterschiedlichen Testebenen erfolgen: Unit-Tests prüfen einzelne Funktionen oder Module, Integrationstests das Zusammenspiel mehrerer Komponenten, und End to End Tests komplette Interaktionen. Für Webanwendungen kommen häufig Tools wie Selenium, Cypress oder Playwright zum Einsatz. Diese

Werkzeuge ermöglichen die Simulation von Benutzerinteraktionen (z. B. Klicks, Formulareingaben, Navigation) und das Verifizieren von UI-Zuständen.
Allerdings ist die Implementierung automatisierter Tests mit initialem Aufwand verbunden. Tests müssen entwickelt, gepflegt und regelmäßig angepasst werden, gerade bei Änderungen an der Benutzeroberfläche. Zudem benötigen automatisierte Tests eine stabile Infrastruktur, die parallele Testdurchführung, Zugriff auf Testdatenbanken und simulierte Benutzerumgebungen unterstützt (Gregory & Crispin, 2015).

Testautomatiserungspyramide

Die Testautomatisierungspyramide ist ein konzeptionelles Modell zur effizienten Verteilung von automatisierten Tests, eingeführt von Mike Cohn (2009).
Sie empfiehlt eine breite Basis aus Unit-Tests, eine mittlere Ebene aus Integrationstests und eine schmale Spitze aus End to End Tests. Ziel ist es, ein optimales Verhältnis zwischen Testtiefe, Ausführungszeit und Wartbarkeit zu erreichen.
Unit-Tests sind am schnellsten, am wenigsten fehleranfällig und liefern unmittelbares Feedback bei Änderungen im Code. Integrationstests sind aufwendiger, da sie Schnittstellen zwischen Systemen testen, liefern aber wichtige Informationen über die Kommunikation der Systemkomponenten. End to End Tests sind die teuerste Testform, denn sie sind langsam, fragil und schwer zu debuggen, dennoch notwendig, um vollständige Geschäftsprozesse zu validieren.
Die Pyramide dient als strategische Orientierungshilfe in der Testplanung. Ein ausgewogenes Verhältnis der Testarten ermöglicht hohe Testabdeckung bei gleichzeitig geringer Wartungslast und schnellen Feedbackzyklen (Fowler, 2018).

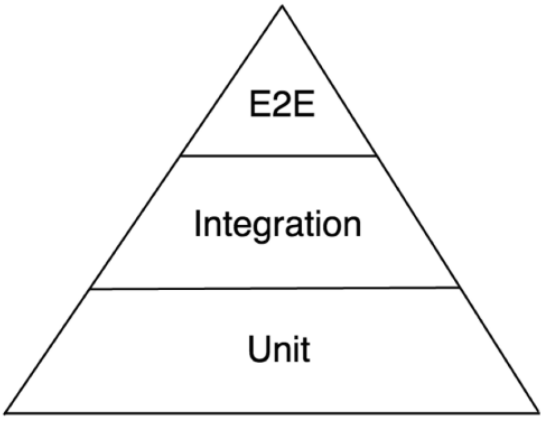

Abbildung 2: Testingpyramide nach Fowler

2.2.2 End-to-End Testing

Wie im vorherigen Kapitel beschrieben worden ist, stellen End-to-End-Tests (E2E) die höchste Teststufe im Softwaretestprozess dar. Sie prüfen die gesamte Anwendung aus der Perspektive eines Endbenutzers. Ziel ist es, sicherzustellen, dass alle Komponenten eines Systems in ihrer Gesamtheit wie vorgesehen funktionieren. Ein typischer E2E-Test könnte etwa den Prozess von der Benutzeranmeldung bis zur Durchführung eines Kaufs in einem Onlineshop abbilden (R. Black, 2012).

Die Tests validieren die korrekte Verarbeitung von Daten und die Interaktion mit externen Services. Außerdem werden visuelle Darstellungen auf der Benutzeroberfläche und das korrekte Zusammenspiel aller Systemschichten geprüft. E2E-Tests sind somit besonders geeignet, um Fehler zu identifizieren, die ausschließlich im Gesamtkontext auftreten.

E2E-Tests werden häufig im Rahmen von User Acceptance Tests (UAT) eingesetzt, bei denen geprüft wird, ob die implementierten Funktionen den Anforderungen des Auftraggebers entsprechen. Sie sind ein wichtiger Bestandteil agiler

Entwicklungsmethoden, insbesondere bei Continuous Delivery und DevOps, da sie eine präzise Prüfung vor der Auslieferung ermöglichen (Gregory & Crispin, 2015).

Typen von End-to-End Tests

E2E-Tests lassen sich in zwei Kategorien unterteilen: horizontale und vertikale Tests. Horizontale E2E-Tests validieren Benutzerworkflows über mehrere Teilsysteme hinweg. Ein Beispiel ist der Prozess von der Registrierung bis zur Zahlungsabwicklung in einem Webshop. Diese Tests prüfen, ob die Integration verschiedener Subsysteme z. B. Frontend, Authentifizierung, Warenkorb und Bezahlung fehlerfrei funktioniert.
Vertikale E2E-Tests fokussieren sich hingegen darauf, einen kompletten Prozess durch alle Schichten eines Systems zu prüfen – von der Benutzeroberfläche über die Business-Logik bis zur Datenbank. Ein praktisches Beispiel hierfür ist eine Online-Überweisung in einer Banking-App: Ein Benutzer gibt die Daten über die Oberfläche ein, die Business-Logik prüft Kontostand und Empfängerdaten, und schließlich wird die Transaktion in der Datenbank gespeichert. Solche Tests umfassen verschiedene Szenarien, z. B. eine Überweisung mit genügend Guthaben und eine mit zu wenig Guthaben, um sicherzustellen, dass das System in allen Schichten korrekt reagiert. Vertikale E2E-Tests haben das Ziel, Abläufe in der gesamten Systemarchitektur zuverlässig zu prüfen und Fehlerquellen in tieferen Ebenen aufzudecken (SmartBear, 2025).

Herausforderungen End-to-End-Testing

E2E-Tests sind in der Praxis mit zahlreichen Herausforderungen verbunden. Der Wartungsaufwand ist hoch, da bereits kleinere Änderungen an der Benutzeroberfläche zu Fehlern führen können. Testskripte müssen regelmäßig angepasst werden, um an die Weiterentwicklung des Systems angepasst zu werden. Zudem sind E2E-Tests langsamer als Unit- oder Integrationstests, was zu längeren Feedback-Zyklen führt.
Ein weiteres Problem ist die Fehlerlokalisierung, denn wenn ein E2E-Test fehlschlägt, ist nicht immer klar, ob das Problem im Frontend, Backend oder an einer Schnittstelle liegt.

Dies erschwert das Debugging und verlängert die Fehlersuche erheblich. Auch die Abhängigkeit von externen Systemen, wie etwa Zahlungsdienstleistern oder E-Mail-Gateways, kann zu instabilen Testergebnissen führen.

Zudem ist die Bereitstellung realistischer Testdaten gerade bei E2E-Tests eine besondere Herausforderung. Da diese Tests vollständige Abläufe unter realistischen Bedingungen abbilden, muss die Testumgebung den Bedingungen einer Live-Umgebung möglichst nahekommen, aber gleichzeitig müssen alle Daten anonymisiert und datenschutzkonform sein. Die Einrichtung und Pflege solcher Umgebungen ist in vielen Projekten mit einem hohen Aufwand verbunden (Sampath & Sprenkle, 2016).

Aus diesen Gründen wird empfohlen, E2E-Tests gezielt auf die wichtigsten und kritischsten Geschäftsprozesse zu beschränken – etwa in Form von „Happy Path"-Tests, die typische und erwartete Abläufe abbilden (Fowler, 2018).

2.2.3 Klassische Testansätze

Manuelle Tests

Manuelle Tests werden als grundlegender Bestandteil des Softwarequalitätsmanagements betrachtet. Die Durchführung erfolgt durch menschliche Tester, die die Anwendung auf vordefinierte Testfälle oder explorativ bedienen. Dabei wird auf Fehler, Unstimmigkeiten oder Usability-Probleme geachtet. Dieses Verfahren basiert vollständig auf menschlicher Interaktion und eignet sich besonders für frühe Entwicklungsphasen, nicht-standardisierte Abläufe oder Anwendungen mit häufigen Änderungen (Sandler & Myers, 2015).

Ein zentrales Merkmal manueller Tests ist ihre Flexibilität. Im Gegensatz zu automatisierten Verfahren können Tester spontan auf unerwartetes Verhalten der Software reagieren. Außerdem können sie verschiedene Wege in der Anwendung erkunden und dabei auch subjektive Kriterien wie Benutzerfreundlichkeit oder visuelle Gestaltung bewerten. Dadurch sind manuelle Tests besonders geeignet für intuitive Tests, GUI-Überprüfungen, User Acceptance Tests (UAT). Darüber hinaus eignen sie sich auch für Systeme, bei denen sich häufig etwas ändert, da ständige Anpassungen der Testskripte sonst zu aufwändig und unwirtschaftlich wären.

Allerdings weisen manuelle Tests auch Nachteile auf. Denn sie sind zeitaufwändig, wenig reproduzierbar und stark von der individuellen Kompetenz der Tester abhängig. Hinzu kommt, dass ihre Skalierbarkeit begrenzt ist, da mit zunehmender Komplexität der Software auch der manuelle Testaufwand exponentiell wächst. Zudem lassen sich manuelle Tests nicht sinnvoll in Continuous Integration (CI)-Pipelines einbinden, wodurch sie für agile oder DevOps-Prozesse nur eingeschränkt geeignet sind (ISTQB, 2018).

Trotz dieser Einschränkungen bleibt der manuelle Test auch im Zeitalter automatisierter Verfahren unverzichtbar. Er stellt die einzige Möglichkeit dar, bestimmte nicht-funktionale Eigenschaften Benutzerfreundlichkeit, Usability oder visuelle Gestaltung verlässlich zu bewerten (Garousi, 2022).

Skriptbasierte Automatisierung

Die skriptbasierte Testautomatisierung ist ein methodischer Ansatz, bei dem Testfälle in Form ausführbarer Programme oder Skripte formuliert und automatisch in der Anwendung ausgeführt werden. Dabei werden Benutzeraktionen, wie das Klicken von Schaltflächen oder das Ausfüllen von Formularfeldern mithilfe von Tools wie Selenium, Playwright oder Cypress simuliert. Danach wird das Verhalten der Anwendung mit vordefinierten Soll-Zuständen verglichen (Garousi, 2022).

Ein entscheidender Vorteil dieses Ansatzes liegt in der Wiederverwendbarkeit: Einmal erstellte Tests können beliebig oft ausgeführt werden, ohne weiteren personellen Aufwand. Zudem lassen sich skriptbasierte Tests gut in moderne CI/CD-Pipelines integrieren, was schnelle Rückmeldungen bei jeder Codeänderung ermöglicht. Dies ist besonders wertvoll für Regressionstests, da sie in kurzen Abständen und mit einer hohen Zuverlässigkeit durchgeführt werden müssen.

Skriptbasierte Tests bieten auch eine hohe Präzision, denn sie dokumentieren exakt, welche Eingaben vorgenommen und welche Systemreaktionen erwartet werden. Darüber hinaus lassen sich mit entsprechenden Frameworks auch Testdaten und simulierte Dienste („Mocks“) integrieren, was die Aussagekraft und Automatisierbarkeit weiter erhöht (ISTQB, 2018).

Allerdings bringt auch dieses Vorgehen auch Herausforderungen mit sich. Eine der größten Schwächen ist die Wartungsanfälligkeit, denn schon kleinere Änderungen an der UI, wie zum Beispiel ein verändertes DOM-Element oder eine CSS-Klasse, können zu einem Fehler des gesamten Tests führen. Diese Beobachtung ist als fragiles Testproblem bekannt. Zudem ist die initiale Erstellung von Testscripts aufwendig und erfordert sowohl Kenntnisse über den fachlichen Kontext als auch technisches Know-how in Programmierung und Testframeworks.

Während skriptbasierte Verfahren heute den Standard für automatisierte Tests darstellen, fehlt ihnen in der Regel die Fähigkeit zur dynamischen Anpassung. Ihre Abläufe sind starr und reagieren nicht auf Veränderungen im Verhalten der Anwendung. Diese Einschränkung bildet den Ausgangspunkt für die Entwicklung moderner, KI-gestützter Testansätze, welche im folgenden Kapitel näher betrachtet werden.

2.2.4 KI-gestützte Testansätze

Begriffseinordnung

Künstliche Intelligenz (KI) im Softwaretesting beschreibt den Einsatz lernfähiger Algorithmen zur Optimierung und Automatisierung von Testaktivitäten. Im Gegensatz zu traditionellen, regelbasierten Verfahren nutzen KI-gestützte Tools strukturierte und unstrukturierte Daten (z. B. Nutzerinteraktionen, Logs, Fehlerhistorien), um daraus Muster zu erkennen um daraus Entscheidungen abzuleiten und sich flexibel an Veränderungen im System anzupassen (Arcuri, 2021).

Während klassische Automatisierungen mit fest programmierten Testskripten laufen, arbeiten KI-basierte Systeme datengetrieben. Sie nutzen dabei Methoden wie maschinelles Lernen oder Mustererkennung, um aus den gesammelten Daten zu lernen. Typisch für solche Systeme sind zum Beispiel die automatische Erstellung von Testfällen, die Priorisierung wichtiger Tests nach Risiko, das Erkennen von Abweichungen sowie die automatische Anpassung bestehender Tests, wenn sich die Anwendung ändert (Self-Healing).

KI-basierte Testautomatisierung ist nicht als Ersatz, sondern als Erweiterung traditioneller Verfahren zu verstehen. Sie adressiert vor allem die Schwächen skriptbasierter Ansätze, wie etwa hohe Wartungskosten, mangelnde Flexibilität und begrenzte Skalierbarkeit (Zhang, 2020). In modernen Continuous-Delivery-Umgebungen gewinnt dieser Ansatz zunehmend an Bedeutung, insbesondere bei Anwendungen, die in viele kleine Module aufgeteilt sind und regelmäßig aktualisiert werden, wie Microservices oder webbasierten Plattformen.

KI-gestützte Testsysteme bieten eine Vielzahl an Funktionen, die weit über die Fähigkeiten klassischer Testautomatisierung hinausgehen. Im Folgenden werden die zentralen Merkmale näher erläutert:

Testfallgenerierung

Durch Analyse realer Nutzungsdaten, Fehlerprotokolle oder Quellcode-Strukturen können Testfälle automatisiert abgeleitet werden. Algorithmen identifizieren typische Interaktionsmuster oder besonders risikobehaftete Pfade und generieren auf dieser Grundlage gezielte Tests. Dabei kommen z. B. Reinforcement Learning oder Clustering-Methoden zum Einsatz (Zhang, 2020).

Testpriorisierung und Optimierung

KI-Modelle werten historische Fehlerdaten, Code-Änderungshäufigkeiten aus, um eine risikobasierte Priorisierung von Testfällen vorzunehmen. So werden besonders fehleranfällige Komponenten zuerst geprüft, was zu einem effizienteren Ressourceneinsatz führt. Dieses Prinzip ist vor allem bei eingeschränkter Testzeit von Vorteil (Martensson, 2022).

Self-Healing

Eine der herausragendsten Eigenschaften vieler KI-basierter Tools ist die Fähigkeit zur automatischen Reparatur instabiler Tests. Wird beispielsweise ein UI-Element umbenannt oder verschoben, erkennt das System dies anhand semantischer oder visueller Ähnlichkeiten und passt das Testskript automatisch an. So wird das „Fragile Test Problem“ erheblich reduziert (Arcuri, 2021).

Anomalieerkennung

KI-gestützte Testsysteme können während des Testens automatisch erkennen, wenn sich eine Software ungewöhnlich verhält. Dazu lernt die KI, wie sich das System normalerweise reagiert, etwa bei Antwortzeiten oder Benutzerinteraktionen. Weicht etwas stark davon ab, wird dies als mögliche Fehlerquelle markiert, auch ohne dass zuvor jeder erwartete Wert genau festgelegt worden ist (Choudhary, 2023)

Lernfähige Testoptimierung

Viele moderne Tools verfügen über lernfähige Mechanismen. Das bedeutet: Mit jeder Testausführung verbessert sich das Modell durch Feedback und Auswertung, etwa durch A/B-Vergleiche oder durch Gegenüberstellung von Ist- und Soll-Verhalten über mehrere Releases hinweg.

Diese Funktionen zeigen: KI-gestützte Testautomatisierung ist nicht nur ein technologischer Fortschritt, sondern ein Paradigmenwechsel im Softwaretestprozess. Durch adaptives

Verhalten, kontinuierliches Lernen und datenbasierte Entscheidungen werden viele Grenzen traditioneller Verfahren überwunden.

Herausforderung

Trotz ihrer Vorteile bringen KI-gestützte Testsysteme auch neue Herausforderungen mit sich. Die Transparenz der Entscheidungssysteme ist begrenzt, denn warum ein KI-Modell einen bestimmten Test empfiehlt oder anpasst, ist nicht immer nachvollziehbar. Zudem ist die Qualität der eingesetzten Daten von zentraler Bedeutung – fehlerhafte Trainingsdaten können zu falschen Testpriorisierungen oder unzuverlässigem Verhalten führen (Arcuri, 2021).

Auch auf organisatorischer Ebene gibt es einige Hürden. Neue Testverfahren müssen erst mal in die bestehenden Abläufe und Tools integriert werden, was oft nicht so einfach ist. Außerdem braucht es ein Umdenken in der bisherigen Teststrategie, was in vielen Unternehmen Zeit braucht. Da es zudem noch an praktischer Erfahrung fehlt, werden KI-basierte Tests aktuell meist nur unterstützend eingesetzt – zum Beispiel zur Optimierung einzelner Schritte oder als Ergänzung zu manuellen Tests, aber noch nicht als vollständiger Ersatz für die klassischen Methoden (Zhang, 2020).

2.2.5 Vergleich beider Ansätze

Ein systematischer Vergleich zwischen klassischen und KI-gestützten Testansätzen zeigt wesentliche Unterschiede in Vorgehen, Automatisierungsgrad, Anpassungsfähigkeit und Skalierbarkeit. Beide Ansätze verfolgen das Ziel, die Qualität von Softwareprodukten durch systematisches Testen zu sichern, unterscheiden sich jedoch in der Art und Weise, wie sie dieses Ziel erreichen.

Automatisierungsgrad und Wartbarkeit

Skriptbasierte Testautomatisierung, wie sie in klassischen Ansätzen verwendet wird, ist vorbestimmt und regelbasiert. Sie beruht auf vordefinierten Testskripten, die exakt

beschreiben, wie die Anwendung getestet werden soll. Dies bietet zwar hohe Präzision und Wiederholbarkeit, führt jedoch zu Wartungsaufwand bei jeder Änderung in der Benutzeroberfläche oder im Geschäftsprozess (Garousi, 2022).
Im Gegensatz dazu ermöglichen KI-gestützte Testverfahren eine flexible Automatisierung. Self-Healing-Mechanismen und maschinelles Lernen reduzieren den manuellen Wartungsaufwand erheblich, indem sie UI-Änderungen automatisch erkennen und darauf reagieren (Martensson, 2022). Dadurch bleibt die Testabdeckung auch bei sich dynamisch entwickelnden Anwendungen stabiler.

Testgenerierung und Abdeckung

Klassische Ansätze erfordern die manuelle Erstellung von Testfällen und -skripten. Die Testabdeckung hängt dabei stark von der Fachkenntnis und Erfahrung der Testenden sowie von den zur Verfügung stehenden Ressourcen ab. In der Praxis führt dies häufig dazu, dass Tests vor allem auf bekannte oder offensichtliche Szenarien abzielen, während Randfälle oder seltene Nutzungsmuster nur unzureichend berücksichtigt werden.
KI-basierte Verfahren bieten hier entscheidende Vorteile, denn sie können automatisch Testfälle aus realen Nutzungsdaten ableiten. Durch die Analyse historischer Fehler oder tatsächlicher Nutzerinteraktionen lassen sich zudem Muster erkennen, die auf kritische Bereiche hinweisen. Diese automatisierte Priorisierung erhöht die Wahrscheinlichkeit, auch unerwartete Fehlerquellen frühzeitig zu identifizieren. Insgesamt kann so eine größere Testtiefe und -breite erreicht werden, ohne dass der manuelle Aufwand proportional steigt (Zhang, 2020).

Transparenz und Kontrolle

Ein Vorteil klassischer Testverfahren liegt in ihrer Nachvollziehbarkeit, denn Testfälle sind explizit formuliert und vollständig kontrollierbar. KI-basierte Systeme hingegen operieren häufig als Black Box, da ihre Entscheidungen, etwa bei der Priorisierung von Tests oder bei

Anpassungen, nicht immer transparent und nachvollziehbar sind. Dies kann die Akzeptanz verringern (Martensson, 2022).

Reifegrad und Integration

Klassische Tools wie Selenium oder Playwright sind seit Jahren etabliert, besitzen große Entwickler-Communities und lassen sich problemlos in bestehende CI/CD-Pipelines integrieren. Auch KI-basierte Tools bieten in der Regel Integrationen für gängige Pipelines. Allerdings erfordert ihre Einführung oft Anpassungen in den bestehenden Prozessen und einen kulturellen Wandel im Team, um das Potenzial automatisierter Entscheidungen und Self-Healing-Funktionen effektiv zu nutzen. (Arcuri, 2021).

2.3 Multi-Criteria Decision Analysis (MCDA)

2.3.1 Entscheidungsunterstützung

Bei komplexen Entscheidungsprozessen, bei denen mehrere – teils konkurrierende – Kriterien zu berücksichtigen sind, hat sich die Multi-Criteria Decision Analysis (MCDA) als bewährte Methode etabliert (Geldermann & Lerche, 2014). Sie ermöglicht eine strukturierte, transparente und nachvollziehbare Bewertung von Entscheidungsalternativen und unterstützt dadurch den Entscheidungsprozess. Bei der Auswahl geeigneter Testverfahren, wie sie im Rahmen dieser Arbeit durchgeführt wird, bietet die MCDA-Methode eine methodische Grundlage, um technische, wirtschaftliche und organisatorische Kriterien systematisch zu bewerten und zu gewichten.

2.3.2 Aufbau und Struktur der MCDA-Matrix

Die MCDA-Matrix ist das zentrale Werkzeug der Multi-Criteria Decision Analysis (MCDA). Sie stellt die Entscheidungsalternativen in den Zeilen den Bewertungskriterien in den Spalten gegenüber. Jede Alternative erhält für jedes Kriterium einen Wert, der angibt, wie gut dieses Kriterium erfüllt wird (Schär, 2018). Die Werte können unterschiedliche Skalenformate haben – zum Beispiel Punkte, Kosten, Nutzen oder normierte Scores. Auf diese Weise lässt sich die Qualität jeder Alternative in Bezug auf alle Kriterien transparent abbilden und vergleichen (Belton & Stewart, 2002).
Ein typischer Aufbau einer MCDA-Matrix könnte zum Beispiel so aussehen: In diesem Beispiel wurde eine Skala von 1 bis 5 gewählt, wobei 1 die schlechteste und 5 die beste Erfüllung eines Kriteriums darstellt.

Alternative/ Kriterium	K1 (1-5)	K2 (1-5)	K3 (1-5)	Gesamtpunktzahl
A1	2	3	2	2,33
A2	1	4	5	3,33

Tabelle 1: Beispiel MCDA-Matrix

Alternative/ Kriterium	j_1	j_2	j_3	Gesamtpunktzahl
i_1	x_{i1j1}	x_{i1j2}	x_{i1j3}	U_1
i_2	x_{i2j1}	x_{i2j2}	x_{i2j3}	U_2

Tabelle 2: Allgemeine MCDA-Matrix

Dabei symbolisiert x_{ij} den Wert, der Alternative i im Hinblick auf Kriterium j erreicht. In der obigen Darstellung wurden die Kriterien gleichgewichtet. Die Gesamtpunktzahl ergibt sich somit als arithmetisches Mittel der Bewertungen über alle Kriterien:

$$U_i = \frac{1}{n} \sum_{j=1}^{n} x_{ij}$$

In vielen realen Anwendungsfällen werden die Kriterien jedoch unterschiedlich gewichtet, da sie eine unterschiedliche Relevanz im Entscheidungsprozess besitzen. In solchen Fällen wird der Gesamtnutzenwert U_i für eine Alternative i nachfolgender gewichteter Formel berechnet:

$$U_i = \sum_{j=1}^{n} w_j \times x_{ij}$$

Dabei bezeichnet:

- x_{ij} den Bewertungswert von Alternative in Bezug auf Kriterium j,
- w_j das Gewicht des Kriterium, wobei $\sum w_j = 1$.

Die Gewichtung kann empirisch (z. B. durch Expertenschätzungen) oder methodisch fundiert erfolgen. Die Anwendung der Formel erlaubt es, die Gesamtbewertung jeder Alternative transparent und nachvollziehbar zu berechnen. Das Ergebnis unterstützt Entscheidungsträger dabei, verschiedene Alternativen objektiv zu vergleichen.

2.3.3 Schritte zur Erstellung und Anwendung der MCDA-Matrix

Die Anwendung der MCDA-Matrix erfolgt in mehreren systematischen Schritten. Diese strukturieren den Entscheidungsprozess und ermöglichen eine nachvollziehbare Grundlage für die Bewertung mehrdimensionaler Alternativen (Geldermann & Lerche, 2014).

1. **Definition des Entscheidungsproblems:**

Zunächst wird das Ziel des Entscheidungsprozesses klar definiert. Teil dieser Arbeit ist die Auswahl eines geeigneten Testverfahrens für ein DMS. Dazu werden die zu bewertenden Alternativen, etwa klassische Testautomatisierung und KI-gestütztes Testing, festgelegt.

2. **Auswahl und Strukturierung der Kriterien:**

Nachfolgend werden relevante Bewertungskriterien identifiziert. Gegebenenfalls erfolgt eine hierarchische Gliederung, etwa in technische, organisatorische und wirtschaftliche Dimensionen. Die Auswahl der Kriterien sollte theorie- und praxisbasiert erfolgen (vgl. Kap. 4.1).

3. **Gewichtung der Kriterien:**

Die Kriterien werden entsprechend ihrer Bedeutung gewichtet. Dies kann durch direkte Bewertung, Rangbildung erfolgen . In dieser Arbeit erfolgt die Gewichtung exemplarisch durch eine Prozentverteilung auf der Grundlage von Experteneinschätzung.

4. **Bewertung der Alternativen:**

Jede Alternative wird für jedes Kriterium anhand einer Skala (z. B. 1–5 Punkte) bewertet. Die Bewertungen basieren auf Literaturanalysen, technischen Eigenschaften und Einschätzungen von Experten.

5. **Sensitivitätsanalyse:**

Im letzten Schritt kann überprüft werden, wie empfindlich das Ergebnis auf Veränderungen bei den Gewichtungen oder Bewertungen reagiert. Dies dient der Überprüfung der Robustheit des Modells. Besonders bei geringen Bewertungsabständen zwischen den Alternativen ist diese Analyse empfehlenswert.

2.3.4 Methodenvielfalt und Auswahl

Die MCDA umfasst eine Vielzahl unterschiedlicher Methoden, darunter etablierte Ansätze wie der Analytic Hierarchy Process (AHP), PROMETHEE oder TOPSIS (Belton & Stewart, 2002). Diese unterscheiden sich hinsichtlich mathematischer Komplexität, Transparenz und Anwendungsdomäne. Für den vorliegenden Anwendungsfall – die vergleichende Bewertung von Testverfahren – wird ein pragmatischer Ansatz in Form eines gewichteten Scoring-Modells verwendet. Diese Variante eignet sich insbesondere für praxisnahe Entscheidungssituationen mit begrenzter Datenlage und ermöglicht eine transparente und nachvollziehbare Bewertung.

2.3.5 Vorteile und Einsatzbereiche der MCDA-Matrix

Ein wesentliches Merkmal der MCDA-Matrix ist die Transparenz, denn Bewertungen, Gewichtungen und Einzelwerte werden explizit sichtbar gemacht. Dadurch wird der Entscheidungsprozess nicht nur nachvollziehbar, sondern auch objektiv überprüfbar (Geldermann & Lerche, 2014).

Ein besonderer Vorteil liegt in der Flexibilität der Methode, denn die Matrix ermöglicht es, sowohl messbare Zahlen (quantitative Kriterien) als auch Einschätzungen (qualitative Kriterien) abzubilden und in einer gemeinsamen Bewertung zusammenzuführen. Daher können technische Merkmale, wirtschaftliche Aspekte und subjektive Einschätzungen in die Entscheidungsfindung einzubeziehen. Zudem erleichtert die MCDA es Teams, sich auf eine

Entscheidung zu einigen, da alle Bewertungsfaktoren transparent gemacht und besprochen werden können (Belton & Stewart, 2002).
Die Einsatzmöglichkeiten sind vielfältig. Zum Einen wird MCDA in der Wirtschaft etwa für Investitionsentscheidungen oder bei der Softwareauswahl eingesetzt. Zum Anderen dient sie im technischen Bereich dazu, IT-Strukturen oder Entwicklungspläne zu bewerten. (Schär, 2018).

Für diese Arbeit ist besonders entscheidend, dass die MCDA neben den technischen auch die wirtschaftlichen Faktoren abbildet. Gerade für kleine und mittlere Unternehmen ist dies hilfreich, da beide Perspektiven für ihre Entscheidungen wichtig sind.

3 Vergleich der Testverfahren

3.1 Traditionelles Testverfahren: Playwright

Playwright ist ein von Microsoft entwickeltes Open-Source-Framework zur browserübergreifenden End-to-End-Testautomatisierung von Webanwendungen. Es unterstützt die automatisierte Interaktion mit Benutzeroberflächen in Chromium-, Firefox- und WebKit-basierten Browsern und deckt damit Nutzungsszenarien ab, wie sie auch in der Praxis vorkommen.

Die Tests werden als Skripte formuliert und ausgeführt, was eine sehr genaue Steuerung der Nutzeraktionen erlaubt, jedoch auch einen höheren Wartungsaufwand mit sich bringt. Dies ist besonders bei dynamischen Anwendungen mit häufigen UI-Änderungen relevant. Durch direkte Ansteuerung der Browser werden Nutzeraktionen wie Klicks, Texteingaben oder Ladezeiten besonders präzise und stabil ausgeführt, was insgesamt eine hohe Performance ermöglicht.

Zu den zentralen Funktionen zählen die Unterstützung mehrerer Programmiersprachen (u. a. TypeScript, Python, Java) sowie die parallele Testausführung. Außerdem bietet Playwright out-of-the-box-Funktionen wie Auto-Waiting, API-Testing und die Möglichkeit, Screenshots oder Videoaufnahmen zu erstellen, um das Debugging zu vereinfachen. Des Weiteren lässt sich Playwright leicht in CI/CD-Pipelines integrieren, wodurch der praktische Einsatz in automatisierten Entwicklungs- und Testprozessen erleichtert wird. Dies ist ein wesentlicher Vorteil für Entwicklerteams. (Microsoft, 2025).

Im Vergleich zu klassischen Tools wie Selenium oder Cypress punktet Playwright mit einer modernen Architektur und mehreren standardmäßig verfügbaren Funktionen. Damit stellt Playwright ein praxisnahes Beispiel für ein klassisches, skriptbasiertes Testwerkzeug dar.

3.2 KI-gestütztes Testverfahren: Mabl

Mabl ist ein kommerzielles, KI-gestütztes Testautomatisierungstool, das sich insbesondere auf End-to-End- und UI-Tests in Cloud-Umgebungen spezialisiert hat. Mabl verwendet intelligente Funktionen wie Self-Healing, das kleinere Änderungen in der

Benutzeroberfläche automatisch erkennt und die Tests entsprechend anpasst. Darüber hinaus erstellt Mabl mithilfe von auto-generierten Testpfaden realistische Interaktionsabläufe des Nutzers und erkennt durch eine Änderungsanalysen, welche bestehenden Tests bei Änderungen im Code oder der UI angepasst werden sollten. Zusätzlich nutzt Mabl maschinelles Lernen, um Tests gezielt zu optimieren und unnötige Testschritte zu reduzieren. Mabl ermöglicht es auch nicht-technischen Nutzern, automatisierte Tests über eine visuelle Oberfläche zu erstellen, was den Zugang für cross-funktionale Teams erleichtert (Mabl, 2025).
Darüber hinaus bietet das Tool eine Integration in gängige DevOps-Umgebungen (z. B. Jira, GitHub, Jenkins) sowie cloudbasiertes Reporting und Testausführung. Die Ergebnisse werden durch KI-gestützte Analysen interpretiert, was insbesondere bei großem Testvolumen eine erhebliche Zeitersparnis verspricht.
Mabl dient in dieser Arbeit exemplarisch als Vertreter eines KI-gestützten Testwerkzeugs.

3.3 Vergleich der Tools

Die beiden ausgewählten Testwerkzeuge – Playwright und Mabl – werden im Folgenden anhand zentraler Merkmale gegenübergestellt. Während Playwright als Open-Source-Framework den klassischen, skriptbasierten Ansatz repräsentiert, steht Mabl für eine moderne, KI-gestützte Lösung mit automatischer Anpassung und Self-Healing-Mechanismen.
Beide Tools lassen sich gut in CI/CD-Pipelines einbinden und decken typische End-to-End-Szenarien ab. Sie unterscheiden sich jedoch deutlich in Bezug auf Wartungsaufwand, Zielgruppe und Automatisierungsgrad.

Die wichtigsten Gemeinsamkeiten und Unterschiede sind in Tabelle X dargestellt.

Kriterium	Playwright (klassisch)	Mabl (KI-gestützt)
Lizenz	Open Source	Kommerziell
Testansatz	Skriptbasiert	KI-basiert
Zielgruppe	Entwickler, QA Engineers	Team übergreifend
Wartung	Manuell, wartungsintensiv bei UI-Änderung	Self-Healing bei UI-Veränderungen
Integration	GitHub, Jenkins, Azure Pipelines	CI/CD, Jira, GitHub, Slack
Testdatenanalyse	Manuell	ML-basiert (z. B. Clustering, Pfadanalysen)
Nutzung ohne Programmierung	Nein	Ja

Tabelle 3: Vergleich Playwright vs. Mabl

3.3.1 Auswahl der Tools

Die Auswahl der beiden Testwerkzeuge Playwright und Mabl ist entlang der zentralen Forschungsfrage dieser Arbeit erfolgt: dem Vergleich klassischer, skriptbasierter mit KI-gestützten Testansätzen. Entscheidend war dabei, methodisch möglichst reine Vertreter beider Paradigmen zu wählen, die zugleich eine hohe praktische Relevanz aufweisen.

Playwright wurde als traditionelles Testwerkzeug ausgewählt, da es aktuell als eines der modernsten und leistungsfähigsten Open-Source-Frameworks für End-to-End-Testing gilt. Das von Microsoft entwickelte Tool ist seit Anfang 2020 öffentlich verfügbar und hat sich innerhalb weniger Jahre eine breite Anwenderbasis aufgebaut. Mit bis zu über 25 Millionen wöchentlichen Downloads (Stand: Mitte 2025) zählt Playwright heute zu den am weitesten verbreiteten Testautomatisierungstools seiner Klasse (npm-stat.com, 2025). In der

Fachliteratur und Community gilt es zunehmend als moderne Alternative zu älteren Frameworks wie Selenium (Qutera, 2025).

Mabl wurde als Gegenstück gewählt, da es zu den wenigen Testwerkzeugen zählt, die vollständig auf KI-basierter Testautomatisierung beruhen. Im Unterschied zu hybriden Ansätzen wie bei Testim oder Functionize integriert Mabl maschinelles Lernen konsequent in alle Kernfunktionen beispielsweise bei der Testfallgenerierung, automatischen Wartung (Self-Healing), Anomalieerkennung und dem Einsatz zur Testvervollständigung. Auch in Bezug auf die Verbreitung gehört Mabl zu den führenden Anbietern im Bereich KI-gestützter Testtools und wird in zahlreichen Unternehmen eingesetzt.

Durch die Auswahl dieser beiden Tools können klassische und KI-basierte Ansätze sauber gegenübergestellt werden. Beide Werkzeuge gelten innerhalb ihres jeweiligen Paradigmas als moderne, leistungsstarke und etablierte Lösungen. Damit bildet die Gegenüberstellung eine realitätsnahe und zugleich systematisch fundierte Grundlage für den weiteren Vergleich im Rahmen dieser Arbeit.

4 Forschungslücke und Forschungsfrage

4.1 Aktueller Forschungsstand

Die Themenbereiche End-to-End-Testing und der Einsatz von Künstlicher Intelligenz im Softwaretest sind einzeln betrachtet gut dokumentiert und in wissenschaftlichen Arbeiten behandelt worden. Eine systematische Verbindung dieser Bereiche in einem praxisorientierten Anwendungskontext insbesondere im Umfeld von Dokumentenmanagementsystemen (DMS) ist jedoch bisher nur unzureichend erforscht.

4.1.1 End-to-End-Testing

Im Bereich des End-to-End-Testings (E2E) liegt der Forschungsschwerpunkt vor allem auf den technischen Herausforderungen dieser Testform. E2E-Tests überprüfen vollständige Geschäftsprozesse über mehrere Systemkomponenten hinweg und gelten als besonders realitätsnah. Laut Black et al. (2012) ist diese Testform essenziell, um die Anforderungen an das System und dem Prozess ganzheitlich zu prüfen. Dabei werden jedoch immer wieder Probleme wie hohe Wartungskosten, Instabilität bei UI-Änderungen sowie komplexe Fehleranalysen thematisiert (Rwemalika et al., 2018).

4.1.2 KI-gestütztes Testing

In den letzten Jahren sind verstärkt KI-basierte Methoden im Softwaretest erforscht worden. Studien wie die von Arcuri (2021). untersuchen unter anderem selbstheilende Testautomatisierung, intelligente Testfallerzeugung oder Priorisierung durch maschinelles Lernen. Die Ergebnisse zeigen, dass KI-Verfahren vielversprechende Ansätze bieten, und Herausforderungen beim Testing wie Skriptpflege, flüchtige Tests und Ressourcenverbrauch beheben. Allerdings ist die Forschung häufig auf prototypische Frameworks oder einzelne Funktionen fokussiert. Ein direkter, strukturierter Vergleich mit traditionellen Methoden insbesondere im realitätsnahen Anwendungskontext bleibt in vielen Arbeiten aus.

4.2 Forschungslücke und Bedarf

Die vorhandene Literatur bildet die Einzelthemen zwar teilweise ab, verknüpft diese jedoch bislang nicht in einem anwendungsorientierten Modell. Insbesondere der direkte methodische Vergleich von traditionellen und KI-gestützten End-to-End-Testverfahren in Bezug auf Wartbarkeit, Aufwandsverteilung, Flexibilität und Fehleranfälligkeit ist wissenschaftlich kaum aufgearbeitet.

Zudem wird der Aspekt der praktischen Entscheidungsunterstützung für KMU kaum berücksichtigt. Gerade kleinere Unternehmen verfügen häufig weder über ausgeprägte Ressourcen noch über spezifisches KI-Know-how. Oft fehlt ihnen ein strukturierter Ansatz, um moderne Testmethoden an ihre genauen Rahmenbedingungen anzupassen und zu bewerten. Gerade im Bereich des Softwaretestings führt die große Auswahl an verfügbaren Tools häufig zu Unsicherheiten bei der Einführung neuer Technologien.

Darüber hinaus mangelt es an anpassbaren Bewertungsmodellen, die technische, wirtschaftliche und organisatorische Kriterien zugleich berücksichtigen. Ein Vergleich mit Hilfe der MCDA-Matrix kann genau hier ansetzen, durch strukturierte Gewichtung und transparente Bewertung lassen sich Stärken und Schwächen der Ansätze nachvollziehbar gegenüberstellen. Bisher existieren keine wissenschaftlichen Studien, die ein solches Modell im Testkontext – insbesondere in Verbindung mit DMS – entwickeln und evaluieren

4.3 Ableitung der Forschungsfrage

Aus der beschriebenen Forschungslücke ergibt sich die zentrale Fragestellung dieser Arbeit:
Wie lassen sich traditionelle und KI-gestützte End-to-End-Testverfahren im Kontext von Dokumentenmanagementsystemen mithilfe einer multikriteriellen Entscheidungsanalyse systematisch vergleichen, um kleinen und mittleren Unternehmen eine Entscheidungsunterstützung zu bieten?
Zur Beantwortung dieser übergeordneten Frage werden im Rahmen dieser Arbeit folgende Teilaspekte untersucht:

- Welche Bewertungskriterien sind im Kontext von Softwaretests für DMS in KMU besonders relevant?
- Wie unterscheiden sich klassische und KI-gestützte Testverfahren in Bezug auf diese Kriterien?
- Wie kann eine MCDA-Matrix entwickelt und eingesetzt werden, um verschiedene Alternativen in einem konkreten Anwendungsszenario zu vergleichen?

Der methodische Rahmen der Arbeit basiert auf dem Design Science Research-Ansatz, der auf die Entwicklung eines anwendungsbezogenen Artefakts – in diesem Fall einer Entscheidungs- und Vergleichsmatrix abzielt. Ziel ist es, einen praxisorientierten Beitrag zu leisten, der Forschung und Anwendung sinnvoll miteinander verknüpft.

5 Entwicklung des Artefakts (Vergleichsmatrix)

5.1 Auswahl der Bewertungskriterien

Die Auswahl geeigneter Bewertungskriterien ist der erste Schritt und somit die Basis der Entwicklung einer MCDA-Matrix. Ziel ist es, eine passende Auswahl an Kriterien zu definieren, die sowohl die technischen als auch die organisatorischen Anforderungen an Testing-Tools im Kontext von End-to-End-Tests abbilden.

Anzahl der Kriterien

Die Anzahl der zu berücksichtigenden Kriterien ist bewusst auf sechs begrenzt worden, denn wie in der Fachliteratur empfohlen, sollte die Anzahl der Kriterien in MCDA-Anwendungen zwischen fünf und neun liegen. Dadurch kann die Komplexität für die Entscheidungsträger reduziert und gleichzeitig eine ausreichend differenzierte Bewertung ermöglicht werden (Belton & Stewart, 2002). Zu viele Kriterien führen nachweislich zu unklaren Bewertungen und erschweren eine einheitliche Einschätzung durch Experten (Geldermann & Lerche, 2014).

Die hier gewählte Anzahl von sechs Kriterien ermöglicht eine valide und zugleich überschaubare Bewertung beider Testwerkzeuge. Sie orientiert sich dabei sowohl an der theoretischen Relevanz als auch an der praktischen Anwendbarkeit im spezifischen Anwendungsfall.

Auswahl der Kriterien

Die Auswahl der Kriterien stützt sich auf Fachliteratur zur Testautomatisierung (Garousi, 2022), Studien zur KI-gestützten Testautomatisierung (Martensson, 2022) sowie auf methodische Empfehlungen zur Softwarebewertung (Zhang, 2020).

Ergänzend wurde die Praxisperspektive berücksichtigt, insbesondere im Hinblick auf typische Herausforderungen kleiner und mittlerer Unternehmen.

Die ausgewählten Kriterien sind: Fehlererkennungsrate, Wartungsaufwand, Benutzerfreundlichkeit, Integrationsaufwand, Testing-Geschwindigkeit und Kosten. Die inhaltliche Definition und Bewertung dieser Kriterien erfolgt im folgenden Abschnitt.

5.2 Definition und Beschreibung der Kriterien

In diesem Abschnitt werden die ausgewählten Kriterien zur Bewertung der Testing-Tools inhaltlich definiert und zudem ihre Bewertungsskalen erläutert. Die Kriterien werden unabhängig von den konkreten Tools beschrieben, damit sie auch in anderen Anwendungskontexten vergleichbar sind. Die Bewertungen werden im Rahmen der MCDA-Matrix später durch individuelle Einschätzungen vorgenommen.

Fehlererkennungsrate

Die Fehlererkennungsrate beschreibt die Fähigkeit eines Testwerkzeugs, relevante Fehler im Zielsystem zuverlässig aufzudecken. Sie ist ein zentraler Maßstab für die Effektivität des Testverfahrens (Sandler & Myers, 2015). Tools mit hoher Fehlererkennungsrate sind in der Lage, auch seltene oder schwer reproduzierbare Fehler aufzudecken. Die Bewertung erfolgt auf einer Skala von 1 (niedrig) bis 5 (hoch).

Wartungsaufwand

Der Wartungsaufwand bezeichnet den Umfang der erforderlichen Anpassungen und Pflegearbeiten an bestehenden Testfällen bei Änderungen am getesteten System (Garousi, 2022). Hoher Wartungsaufwand entsteht typischerweise durch fehleranfällige, wenig wiederverwendbare Testskripte oder fehlende Self-Healing-Funktionen. Die Bewertung erfolgt ebenfalls auf einer Skala von 1 (sehr hoch) bis 5 (sehr gering).

Benutzerfreundlichkeit

Dieses Kriterium bezieht sich auf die subjektive Bedienbarkeit und Zugänglichkeit des Testing-Tools für unterschiedliche Nutzergruppen (ISTQB, 2018). Dazu zählen Aspekte wie GUI-Design, Dokumentation, Einarbeitungszeit und die Notwendigkeit programmatischer Vorkenntnisse. Eine hohe Benutzerfreundlichkeit liegt vor, wenn auch fachfremde oder nicht-technische Nutzer das Tool effizient bedienen können. Skala: 1 (schlecht nutzbar) bis 5 (sehr nutzerfreundlich).

Integrationsaufwand

Der Integrationsaufwand beschreibt, wie einfach ein Testwerkzeug in bestehende Entwicklungs-, Build- und Deploymentprozesse eingebunden werden kann. Dies umfasst Schnittstellen zu CI/CD-Tools, Testmanagementsystemen oder Versionskontrolle (Martensson, 2022). Eine hohe Integration erleichtert den automatisierten Betrieb und steigert die Testfrequenz. Skala: 1 (sehr aufwendig) bis 5 (sehr einfach).

Testing-Geschwindigkeit

Die Testing-Geschwindigkeit bezieht sich auf die Dauer, die zur Durchführung eines vollständigen Testlaufs benötigt wird, von der Testinitialisierung über die Ausführung bis hin zur Auswertung. Sie beeinflusst maßgeblich die Effizienz iterativer Entwicklungszyklen. Werkzeuge mit paralleler Testausführung, intelligenter Testpriorisierung oder Cloud-Execution bieten in der Regel höhere Geschwindigkeit (Zhang, 2020). Bewertung: 1 (langsam) bis 5 (sehr schnell).

Kosten

Das Kriterium Kosten umfasst die Gesamtkosten des Tools über seinen Lebenszyklus hinweg. Dazu zählen Lizenzgebühren, Hardware- oder Cloudkosten, Schulungsaufwand sowie Wartung und Support (Geldermann & Lerche, 2014). Bewertungsskala: 1 (sehr teuer) bis 5 (sehr kosteneffizient).

Die gewählte Skala von 1 bis 5 ermöglicht eine differenzierte, aber trotzdem gut handhabbare Einschätzung. Die Werte sind für die MCDA-Methode geeignet, da sie ordinal interpretierbar und vergleichbar sind (Belton & Stewart, 2002).

5.3 Struktur und Inhalt der MCDA-Matrix

Die Multi-Criteria Decision Analysis (MCDA)-Matrix stellt das zentrale Artefakt dieser Masterarbeit dar. Die MCDA-Matrix fasst die Bewertung der zuvor definierten Kriterien zusammen, um Playwright und Mabl strukturiert zu vergleichen. Dadurch wird eine nachvollziehbare Bewertung ermöglicht.
Die Bewertung der beiden Tools basiert auf den in Kapitel 4.2 definierten Skalen (1 = sehr schwach bis 5 = sehr stark).
Dabei sind sowohl persönliche Erfahrungswerte aus Projekten als auch Erkenntnisse aus der Fachliteratur eingeflossen.
Die folgende Tabelle zeigt die Bewertungsmatrix (noch ohne Gewichtung):

Kriterium	Playwright	Mabl
Fehlererkennungsrate	3	4
Wartungsaufwand	3	5
Benutzerfreundlichkeit	1	5
Integrationsaufwand	4	5
Testing-Geschwindigkeit	4	4
Kosten	5	1

Tabelle 4: MCDA-Matrix mit Bewertungen

Im Folgenden werden die jeweiligen Bewertungen der MCDA-Matrix detailliert erläutert.

Im Kriterium **Fehlererkennungsrate** wurde Mabl mit 4 Punkten bewertet, während Playwright 3 Punkte erhielt. Die höhere Bewertung für Mabl resultiert aus der Fähigkeit, nicht nur auf vordefinierte Testskripte zurückzugreifen, sondern zusätzlich maschinelles Lernen einzusetzen, um unerwartetes Systemverhalten zu erkennen. In der Praxis zeigte sich, dass Mabl insbesondere durch Algorithmen, die ungewöhnliche Abweichungen erkennen, in der Lage war, UI-Inkonsistenzen, untypische Ladezeiten oder unerwartete Übergänge zu finden, ohne dass diese explizit gescriptet worden sind. Playwright hingegen bietet ein

präzises, aber rein regelbasiertes Fehlererkennungssystem, bei dem nur die im Code hinterlegten Soll-Ist-Abgleiche geprüft werden. Dies limitiert die Fehlerabdeckung auf das vorhandene Wissen und die Erfahrung der Testentwickelnden und kann bei regressiven Tests zu blinden Flecken führen.

Der **Wartungsaufwand** zeigt die deutlichsten Unterschiede. Mabl erhielt hier die Maximalbewertung (5), da es über robuste Self-Healing-Mechanismen verfügt, die UI-Elemente auch nach Änderungen wiedererkennen können. Diese Funktion basiert auf einer Kombination aus Selektorhistorien, semantischem Matching und visuellem Vergleich. Die Praxiserfahrung hat gezeigt, dass sich der Aufwand zur Anpassung von Tests nach UI-Änderungen dadurch erheblich reduziert. Bei einer neuen Version eines DMS-Systems sind keine manuellen Anpassungen erforderlich gewesen. Playwright ist mit 3 Punkten bewertet worden. Zwar bietet Playwright durch seine klare Skriptstruktur eine hohe Kontrolle, verfügt jedoch über keine integrierten Self-Healing-Mechanismen. Es kann Elemente zwar auch über Textinhalte finden, was die Abhängigkeit vom HTML- oder CSS-Code reduziert, dennoch sind bei Änderungen an Strukturen, Klassen oder IDs häufig manuelle Anpassungen nötig, was zu zusätzlichem Debugging-Aufwand führen kann.

Im Kriterium **Benutzerfreundlichkeit** hat Mabl 5 Punkte erhalten. Entscheidende Faktoren dafür sind die grafische Benutzeroberfläche mit Drag-and-Drop-Funktionalität, integrierte Testpfad-Vorschläge, integrierte Aufzeichnung von Testschritten sowie eine umfassende Dokumentation gewesen. Besonders hervorzuheben ist, dass auch nicht-technische Mitarbeitende eigenständig einfache Testszenarien anlegen können, was die Akzeptanz im Unternehmen erhöht und die Zusammenarbeit zwischen Entwicklung und Fachbereich erleichtert. Playwright ist mit 1 Punkt bewertet worden. Zwar ist es gut dokumentiert und bietet moderne APIs, erfordert jedoch Programmierkenntnisse in JavaScript oder Python. Es gibt eine Funktion zur Aufzeichnung von Tests, bei der Nutzer Klicks und Eingaben ausführen und daraus automatisch ein Skript generiert wird. In der Praxis zeigt sich jedoch, dass selbst bei einfachen Tests die Generierung oft unvollständig oder fehleranfällig ist, sodass manuelle Nacharbeit nötig geworden ist. Der Einarbeitungsaufwand für Personen

ohne technische Vorbildung ist daher hoch, und die Fehlersuche bei fehlschlagenen Tests ist oft zeitaufwendig.

Das Kriterium **Integrationsaufwand** wurde bei Mabl mit 5 bewertet. Mabl verfügt über bereitgestellte Integrationen mit gängigen DevOps-Tools wie GitHub, GitLab, Jenkins, Bitbucket, Azure DevOps und Jira. Die Einrichtung erfolgt über eine webbasierte Oberfläche ohne Codeanpassung und wird durch Vorlagen unterstützt. Playwright erhielt 4 Punkte. Es ist vollständig CI/CD-kompatibel und lässt sich problemlos in moderne Pipelines integrieren, erfordert jedoch bei Anbindungen an externe Tools wie Jira oder Slack häufig eigene Skripte oder Drittanbieter-Plugins. Für technisch erfahrene Teams ist dies unproblematisch, für heterogene Teams kann es jedoch eine Einstiegshürde darstellen.
Im Kriterium **Testing-Geschwindigkeit** erhielten sowohl Mabl als auch Playwright eine Bewertung von 4 Punkten.
Die Grundlage dieser Einschätzung haben drei Testdurchläufe mit identischen Testfällen in einer Umgebung aus der Praxis gebildet, wie sie im Alltag auch genutzt werden. Für Playwright sind dabei Ausführungszeiten von 15,6 s, 15,6 s und 15,2 s gemessen worden. Mabl zeigt hierbei mit 14,5 s, 14,8 s und 15,2 s leicht geringere Werte, wobei die Differenz insgesamt als marginal zu bewerten ist. Die Ergebnisse der Testzeiten sind im Anhang zu sehen.
Die erzielten Ergebnisse verdeutlichen, dass beide Werkzeuge eine insgesamt sehr gute Performance aufweisen und in der Lage sind, automatisierte Tests innerhalb kurzer Zeit stabil und reproduzierbar auszuführen. Zwar zeigt Mabl tendenziell etwas kürzere Laufzeiten, diese liegen jedoch im Bereich von unter einer Sekunde Unterschied und sind im praktischen Einsatz nicht signifikant.
Die Bewertung mit jeweils 4 Punkten spiegelt daher eine ausgewogene Einschätzung aus der Praxis wider, die auf real durchgeführten Tests basiert. Sie berücksichtigt sowohl die gemessenen Zeiten als auch die Konstanz der Ausführungen. Beide Tools erfüllen damit die Anforderungen an eine effiziente Testdurchführung in typischen Anwendungsfällen.

Die **Kostenbewertung** der betrachteten Testwerkzeuge basiert auf dem Total-Cost-of-Ownership-Ansatz (TCO), wie er in der MCDA-Literatur als Standardmethode empfohlen wird (Belton & Stewart, 2002). Ein Betrachtungszeitraum von drei Jahren bildet dabei sowohl einmalige Einführungskosten als auch laufende Betriebs- und Wartungskosten realistisch ab (Bichachi, 2025). Für die Eigenentwicklung auf Basis von Playwright ist ein Aufwand von insgesamt 250 Stunden über den gesamten Betrachtungszeitraum angesetzt worden. Zur Bewertung ist ein durchschnittlicher Stundensatz von 85 € definiert worden. Dieser Wert orientiert sich am aktuellen Marktdurchschnitt für Softwareentwickler im deutschsprachigen Raum, wie er im GULP Freelancer-Kompass 2025 für Junior-Entwickler ausgewiesen ist (Gulp, 2025). Daraus ergibt sich ein Gesamtaufwand von 21.250 €, wobei keine zusätzlichen Lizenz- oder Infrastrukturkosten anfallen. Demgegenüber stehen die reinen Lizenzkosten für das KI-gestützte Tool Mabl bei mindestens 20.000 $ pro Jahr (entspricht rund 18.500 € jährlich; Stand 2025). Für drei Jahre ergibt sich damit ein Fixkostenblock von 55.500 €, wobei variable Zusatzkosten für weitere Nutzer oder steigendes Testvolumen nicht enthalten sind. Zusätzlich entsteht ein initialer Aufwand zur Testmodellierung, der trotz der intuitiven Benutzeroberfläche nicht entfällt. Selbst bei einer konservativen Schätzung von 50–100 Stunden ergeben sich daraus zusätzliche Kosten in Höhe von 4.250 € bis 8.500 €, ebenfalls auf Basis des oben genannten Stundensatzes. Auf Grundlage dieser wirtschaftlichen Bewertung ist im Rahmen der MCDA-Matrix das Kriterium „Kosten“ mit 5 Punkten für Playwright und einem Punkt für Mabl bewertet worden. Die hohe Bewertung für Playwright reflektiert die signifikant geringeren Gesamtkosten sowie den Wegfall wiederkehrender Lizenzgebühren. Die niedrigere Bewertung für Mabl berücksichtigt die hohen laufenden Kosten trotz technischer Vorteile in anderen Bereichen.

Insgesamt zeigt sich, dass Mabl vor allem bei nicht-funktionalen Kriterien, wie Wartbarkeit, Benutzerfreundlichkeit und Integration, klare Vorteile gegenüber Playwright aufweist. Playwright punktet hingegen bei technischen Aspekten wie der Kostenstruktur und der direkten Kontrollierbarkeit. Die MCDA-Matrix bietet somit ein differenziertes Bewertungsinstrument, das die Stärken und Schwächen der Tools im jeweiligen

Anwendungskontext sichtbar macht. In Kapitel 5 erfolgt auf dieser Grundlage eine gewichtete Bewertung und die Anwendung auf einen konkreten DMS-Anwendungsfall.

6 Evaluation und Anwendung in der Praxis

6.1 Beschreibung des Praxispartners

Die im Rahmen dieser Arbeit untersuchte Fall Anwendung erfolgt in Kooperation mit der aktivweb System- und Datentechnik GmbH, einem spezialisierten Anbieter von Dokumentenmanagementsystemen (DMS) mit Sitz in Deutschland. Das Unternehmen entwickelt und vertreibt seit mehreren Jahren das Dokumentenmanagement System „Docbox“. Der Einsatz der Docbox ist auf Unternehmen verschiedenster Größen und Branchen ausgerichtet. Die Software kommt sowohl in Industrie- und Dienstleistungsunternehmen als auch in sensiblen und regulierten Sektoren wie dem Gesundheitswesen, dem Finanz- und Steuerbereich sowie in öffentlichen Verwaltungen zum Einsatz. Besonders hervorzuheben ist die Integration von Schnittstellen zu branchenspezifischen Anwendungen, wie etwa zu DATEV, was den Einsatz in Steuerkanzleien und Rechtsanwaltsbüros erleichtert. Die hohe Anpassungsfähigkeit an individuelle Geschäftsprozesse macht die Lösung sowohl für kleine Unternehmen als auch für mittlere und große Organisationen attraktiv. Technologisch basiert die Docbox auf einem modularen System, das sowohl On-Premise als auch cloudbasiert betrieben werden kann. Das System liefert die typischen DMS-Funktionalitäten wie revisionssichere Archivierung, Workflow-Management, Benutzer- und Rechteverwaltung sowie strukturierte Volltextsuche. Ergänzend bietet das System Möglichkeiten zur Integration mit externen IT-Systemen wie ERP- oder CRM-Lösungen und unterstützt rechtliche Anforderungen wie z.B. die DSGVO. Das Unternehmen verfolgt einen praxisorientierten Entwicklungsansatz, bei dem Kundenanforderungen direkt in die Weiterentwicklung der Software einfließen. Diese enge Verzahnung von Produktentwicklung und operativer Anwendung bildet die Grundlage für die in dieser Arbeit durchgeführte Evaluierung von Testautomatisierungsmöglichkeiten im DMS-Umfeld.

6.2 Use Case

Die aktivweb System- und Datentechnik GmbH betreibt seit vielen Jahren eine eigenentwickelte DMS-Lösung, die kontinuierlich weiterentwickelt und kundenindividuell angepasst wird. Die Sicherstellung der Softwarequalität erfolgt dabei bislang vollständig manuell. Tests sind von Entwicklern durchgeführt worden und eine strukturierte Testautomatisierung ist nicht implementiert gewesen.
Dieses Vorgehen ist typisch für viele mittelständische Unternehmen im Softwarebereich, insbesondere wenn branchenspezifische Anforderungen, individuelle Kundenkonfigurationen und enge Releasezyklen zusammentreffen, wie es bei Docbox der Fall ist. In der Praxis hat das manuelle Testen zunehmend zu Herausforderungen in Bezug auf Reproduzierbarkeit**,** Testabdeckung und Ressourceneinsatz geführt. So hat zentrale Geschäftsprozesse wie z. B. Dokumentenimport, Rechtevergabe oder Archivierungsabläufe regelmäßig erneut vollständig manuell prüfen müssen, insbesondere bei größeren Updates oder Infrastrukturänderungen.

Hinzu kommen Skalierungsprobleme, denn mit wachsender Funktionsvielfalt der Docbox-Software ist die manuelle Testabdeckung immer aufwendiger, fehleranfälliger und damit auch zeitintensiver geworden. Damit steigt auch das Risiko von Regressionsfehlern, da nicht alle Abhängigkeiten zwischen Modulen bei Änderungen zuverlässig nachverfolgt und getestet werden können. Gerade im DMS-Umfeld, das durch strenge rechtliche Anforderungen geprägt ist, stellt dies ein Risiko dar.

Aus diesen Gründen hat sich das Unternehmen zur systematischen Evaluierung eines geeigneten Testautomatisierungsansatzes entschlossen. Hierbei ist es das Ziel gewesen, künftig sowohl die Testtiefe als auch die Testeffizienz zu erhöhen. Im Rahmen dieser Arbeit ist daher eine vergleichende Bewertung zweier Automatisierungsansätze vorgenommen worden, einerseits das klassische, skriptbasierte Testing mit Playwright und andererseits ein KI-gestützter Ansatz mit dem Cloud-Tool Mabl.

Die zuvor in Kapitel 4 entwickelte MCDA-Matrix dient hierbei als methodische Grundlage. Im folgenden Kapitel werden die Ergebnisse dieser Anwendung im Detail analysiert und im Hinblick auf den spezifischen Use Case der Docbox interpretiert.

6.3 Anwendung der Matrix in der Praxis

Nachdem in Kapitel 4 die MCDA-Matrix entwickelt worden ist, erfolgt nun die Gewichtung der Kriterien durch den Praxispartner Docbox. Dadurch wird die Relevanz der einzelnen Kriterien im konkreten Unternehmenskontext aufgezeigt und trägt dazu bei, eine individuell angepasste Entscheidung zu ermöglichen.

Die Gewichtung ist durch einen erfahrenen internen Entwickler vorgenommen worden, der die Anforderungen an Softwaretests im DMS-Kontext aus langjähriger Praxis kennt. Dabei sind den sechs Kriterien prozentuale Gewichtungen zugeordnet worden, die deren Bedeutung für die Auswahl eines geeigneten Testwerkzeugs aus Sicht von Docbox widerspiegeln. Die finale Gewichtung stellt sich wie folgt dar:

Kriterium	Gewichtung
Fehlererkennungsrate	0,25
Wartungsaufwand	0,15
Benutzerfreundlichkeit	0,05
Integrationsaufwand	0,15
Testing-Geschwindigkeit	0,05
Kosten	0,35

Tabelle 5: Gewichtung der MCDA-Matrix am Beispiel DOCBOX

Diese Gewichtung verdeutlicht, dass wirtschaftliche und betriebliche Faktoren für Docbox eine besonders hohe Relevanz besitzen. Mit einem Gewicht von 35 % wurde dem Kriterium Kosten die höchste Bedeutung zugemessen, gefolgt von der Fehlererkennungsrate mit 25% sowie Wartungsaufwand und Integrationsaufwand mit jeweils 15 %. Technisch orientierte

Kriterien wie Benutzerfreundlichkeit und Testing-Geschwindigkeit spielen hingegen eine untergeordnete Rolle, da das Unternehmen auf intern vorhandene Entwicklerressourcen zurückgreifen kann.

Bei der Berechnung des Gesamtnutzens sind die Scores der beiden Testwerkzeuge mit den entsprechenden Gewichtungen multipliziert worden. Die Berechnung erfolgt nach dem klassischen MCDA-Prinzip, da sich der Endwert ergibt sich aus der Summe der gewichteten Einzelkriterien (Gewichtung x Bewertung). Die Ergebnisse sind in der folgenden Tabelle zusammengefasst:

Kriterium	Gewichtung	Playwright	Playwright Score	Mabl	Mabl Score
Fehlererkennungsrate	0,25	3	0,75	4	1
Wartungsaufwand	0,15	3	0,45	5	0,75
Benutzerfreundlichkeit	0,05	1	0,05	5	0,25
Integrationsaufwand	0,15	4	0,60	5	0,75
Testing-Geschwindigkeit	0,05	4	0,20	4	0,2
Kosten	0,35	5	1,75	1	0,35
SUMME	**1**		**3,80**		**3,3**

Tabelle 6: Gesamte MCDA-Matrix am Beispiel DOCBOX

Die Berechnung zeigt, dass Playwright mit einem Gesamtwert von 3,8 gegenüber 3,3 für Mabl besser bewertet worden ist. Grund hierfür ist die hohe Bewertung von Playwright im stark gewichteten Kriterium der Kosten. Außerdem trägt auch das solide Abschneiden in den Kriterien Integrationsaufwand und Testing-Geschwindigkeit dazu bei. Die Stärken von Mabl in Bezug auf Benutzerfreundlichkeit und Wartungsaufwand haben den Kostenvorteil von Playwright nicht ausgleichen können.

6.3.1 Sensitivitätsanalyse

Die Sensitivitätsanalyse wird durchgeführt, um die Robustheit der im Rahmen der MCDA ermittelten Ergebnisse zu überprüfen. Ziel ist es, aufzuzeigen, ob sich die Rangfolge der betrachteten Testautomatisierungslösungen bereits bei moderaten Änderungen der Annahmen wesentlich verschiebt, oder ob die Empfehlung auch unter abweichenden

Rahmenbedingungen stabil bleibt. Dies entspricht der Empfehlung von Fachliteratur, welche besagt, dass bei Nutzwertanalysen Unsicherheiten berücksichtigt werden müssen (Belton & Stewart, 2002).

Für die Analyse sind drei realistische Szenarien definiert worden, die zentrale Einflussgrößen gezielt variieren:

- **Szenario A:** Die Gewichtung der Fehlererkennungsrate ist von 25 % auf 35 % erhöht worden, um die hohe Bedeutung der Softwarequalität besser zu verdeutlichen.
- **Szenario B:** Die Bewertung des Wartungsaufwands von Playwright ist von 3 auf 2 Punkte reduziert worden, um potenzielle Mehraufwände bei der langfristigen Pflege und Anpassung zu berücksichtigen.
- **Szenario C:** Die Kostenbewertung für Playwright ist von 5 auf 4 Punkte reduziert worden, um versteckte Kosten durch Schulungs- und Integrationsaufwand abzubilden.
- **Szenario D:** Zusätzlich ist ein Szenario definiert worden, in dem bei Mabl unerwartet günstigere Nutzungskosten auftreten könnten (z. B. durch Rabattmodelle, sinkende Lizenzgebühren oder interne Kostensenkungseffekte). Um dies abzubilden, wurde die Kostenbewertung von Mabl von 1 auf 2 Punkte erhöht.

Für alle Szenarien ist nun der Gesamtwert neu berechnet worden. Um die höhere Bewertung der Fehlererkennungsrate auszugleichen, ist im Szenario A die Gewichtung aller anderen Kriterien um 2% verringert worden. Somit liegt die Summe der Gewichtungen nach wie vor bei 100%.

Szenario	Playwright	Mabl
Basis Szenario	3,8	3,3
Szenario A	3,76	3,3
Szenario B	3,65	3,3
Szenario C	3,45	3,3
Szenario D	3,8	3,65

Tabelle 7: Sensivitätsanalyse der MCDA-Matrix

Die Ergebnisse der Sensitivitätsanalyse zeigen, dass Playwright auch bei realistischen Anpassungen einzelner Einflussgrößen die bevorzugte Lösung bleibt. Anpassungen der Fehlererkennungsrate, des Wartungsaufwands und der Kosten führen zwar zu einer Reduktion des Gesamtnutzenwerts, jedoch ohne Einnahme der Spitzenposition durch Mabl. Dies unterstreicht die Robustheit der Empfehlung zugunsten von Playwright und macht zugleich deutlich, dass zentrale Faktoren wie Wartungsaufwand, Qualitätssicherung und Lizenzkosten in der Praxis regelmäßig überprüft und gegebenenfalls angepasst werden sollten.
Die detaillierten Berechnungen der Teilnutzwerte pro Szenario sind im Anhang dargestellt.

6.4 Einschätzung durch den Praxispartner

Im Nachgang der Anwendung der MCDA-Matrix ist ein abschließendes Reflexionsgespräch mit dem Praxispartner durchgeführt worden. Ziel ist es gewesen, die Praxistauglichkeit des Bewertungsmodells zu überprüfen und die Entscheidung im Licht der tatsächlichen Unternehmenssituation kritisch einzuordnen. Seitens des Praxispartners ist die Bewertungsmethode als sehr hilfreich und praxisnah eingeschätzt worden. Insbesondere die strukturierte Herangehensweise der MCDA-Matrix ist positiv hervorgehoben worden. Durch die transparente Gegenüberstellung von Bewertungskriterien und die explizite Gewichtung kann das Unternehmen nun nachvollziehbar abbilden, welche Anforderungen für sie priorisiert sind und welche Kompromisse mit jeder Alternative einhergehen.

Der Praxispartner bestätigt, dass das Ergebnis der Matrix die realen Entscheidungsmuster im Unternehmen sehr gut widerspiegelt. Zwar ist das KI-gestützte Tool Mabl in mehreren Kriterien, besonders in Bezug auf Wartungsaufwand und Benutzerfreundlichkeit, als technisch überlegen erkannt, jedoch stellte sich im Abgleich mit den unternehmensspezifischen Rahmenbedingungen klar heraus, dass die langfristigen Kosten und der erforderliche Schulungsaufwand für eine KI-basierte Lösung als zu hoch eingeschätzt werden. Wie vom Praxispartner formuliert, sei der Einsatz eines KI-Werkzeugs „methodisch interessant, aber ökonomisch derzeit nicht vertretbar." Besonders hervorzuheben ist, dass die Matrix die Entscheidungsfindung beschleunigt und fundiert hat**,** ohne ein bestimmtes Ergebnis vorzugeben. Das Modell wurde intern auch genutzt, um die Entscheidung gegenüber anderen Stakeholdern, gerade auf Managementebene, zu begründen. Die methodische Klarheit und Nachvollziehbarkeit wird dabei als entscheidender Mehrwert gesehen. Die Empfehlung aus der MCDA-Matrix ist letztlich eins zu eins in die Praxis umgesetzt worden, denn Docbox hat sich entschieden, zukünftig auf den klassischen, skriptbasierten Testansatz mit Playwright zu setzen. Die initialen Implementierungen sind bereits erfolgt, erste Testläufe sind erfolgreich in bestehende Entwicklungsprozesse integriert worden.

Damit kann festgehalten werden, dass das in dieser Arbeit entwickelte Bewertungsmodell nicht nur theoretisch, sondern auch praktisch einsetzbar und entscheidungsrelevant ist. Aus Sicht des Unternehmens stellt die MCDA-Matrix ein wertvolles Instrument zur Toolbewertung und Investitionsentscheidung dar, welches künftig auch in anderen Kontexten zum Einsatz kommen könnte.

7 Fazit und Ausblick

7.1 Zusammenfassung der Ergebnisse

Das Ziel dieser Arbeit ist es gewesen, eine allgemeine und praxisnahe Entscheidungshilfe für Softwareunternehmen zu entwickeln, die sie dabei unterstützt, den für sie geeigneten Testansatz zu finden. Dieses Bewertungsmodell ist im Rahmen eines konkreten Anwendungsfalls bei der Firma Docbox durchgeführt worden, um zu prüfen, ob es auch in der Praxis relevant ist.

Im ersten Teil der Arbeit ist zunächst die Anforderungen an Testautomatisierung im Allgemeinen und speziell im DMS-Umfeld untersucht worden. Darüber hinaus ist klassische und KI-gestützte Verfahren beschrieben worden. Auf dieser Basis können Kriterien für die MCDA-Matrix abgeleitet werden, die sowohl technische als auch wirtschaftliche Aspekte berücksichtigen. Im nächsten Schritt ist die Matrix mit Bewertungen gefüllt worden, indem beide Werkzeuge in allen Kriterien mit Punkten von eins bis fünf eingeschätzt worden sind. Um den praktischen Nutzen der Matrix zu testen, ist die entwickelte MCDA-Matrix bei Docbox eingesetzt worden, um die beiden Testwerkzeuge Playwright und Mabl vergleichbar zu bewerten. Die Gewichtung der Kriterien hat ein erfahrener Entwickler aus dem Unternehmen vorgenommen, um sicherzustellen, dass die Bewertung die strategischen Prioritäten von Docbox widerspiegelt.

Die Analyse zeigt, dass Playwright vor allem durch seine Kostenvorteile überzeugt hat, während Mabl seine Stärken in anderen Bereichen nicht ausreichend ausspielen konnte.

Der Praxispartner bewertet die MCDA-Matrix insgesamt als hilfreiches Instrument um Vor- und Nachteile transparent zu machen und die Entscheidung im Unternehmen nachvollziehbar zu kommunizieren. Letztlich ist die Empfehlung der Analyse umgesetzt worden, indem sich Docbox für Playwright entschieden hat.

Die Arbeit zeigt somit, dass das entwickelte Artefakt in der Praxis funktioniert und eine wertvolle Unterstützung bei komplexen Technologieentscheidungen bieten kann. Es lässt sich auch auf vergleichbare Fragestellungen in anderen Organisationen übertragen.

7.2 Bewertung des Vorgehens

Rückblickend hat sich das gewählte Vorgehen mit der MCDA-Matrix als gut umsetzbar und praxisnah erwiesen. Besonders hilfreich ist gewesen, dass sich die verschiedenen Kriterien in einer klaren Struktur gegenüberstellen lassen und dadurch verschiedene Faktoren berücksichtigt werden konnten. Für den Praxispartner ist es von Vorteil gewesen, dass die Gewichtung flexibel an die eigenen Prioritäten angepasst werden kann, denn dadurch ist eine realistische Bewertung und demnach eine nachvollziehbare Entscheidung möglich.
Gleichzeitig ist deutlich geworden, dass bestimmte Aspekte wie Benutzerfreundlichkeit oder Wartungsaufwand schwer vollständig objektiv messbar sind, somit sind an diesen Stellen praktische Erfahrungen und Einschätzungen gefragt.
Das ist zwar in der Praxis oft unvermeidbar, zeigt aber auch die Grenze einer solchen Methode.
Ein weiterer wichtiger Erfahrungswert ist es gewesen, dass nicht zu viele Kriterien aufgenommen werden sollten. Es wäre zwar möglich gewesen, noch detailliertere Faktoren zu betrachten, doch dies hätte die Übersichtlichkeit deutlich erschwert. Die Entscheidung, sich auf sechs Kernkriterien zu konzentrieren, kann demnach als guter Kompromiss zwischen Genauigkeit und Praktikabilität verstanden werden.
Insgesamt lässt sich sagen, dass die MCDA-Matrix ein wertvolles Werkzeug sein kann, um gerade in mittelständischen Unternehmen komplexe Technologieentscheidungen transparenter zu machen. Die Methode hilft dabei, Argumente strukturiert darzustellen und intern zu vermitteln. Entscheidend ist jedoch, dass die Bewertung gut vorbereitet wird, die Kriterien sauber definiert sind und die Einschätzungen realistisch bleiben. Unter diesen Bedingungen würde ich dieses Vorgehen auch in anderen Projekten empfehlen.

7.3 Empfehlungen für weitere Forschung und Praxis

Die Ergebnisse dieser Arbeit zeigen, dass die Anwendung einer MCDA-Matrix zur Bewertung von Testautomatisierungstools einen klaren Mehrwert bieten kann. Die strukturierte Bewertung ermöglicht es Unternehmen, komplexe Alternativen anhand klar definierter und gewichteter Kriterien zu vergleichen und dabei sowohl technische Leistungsmerkmale als auch wirtschaftliche und organisatorische Rahmenbedingungen zu berücksichtigen. Daraus lassen sich verschiedene Implikationen für die praktische Umsetzung sowie Empfehlungen für die weiterführende Forschung ableiten.

Für die Praxis empfiehlt sich insbesondere für kleine und mittlere Unternehmen (KMU), die häufig mit begrenzten Ressourcen und unstrukturierten Auswahlprozessen konfrontiert sind, der Einsatz strukturierter Entscheidungsmodelle wie der MCDA. Durch die Möglichkeit, unternehmensspezifische Gewichtungen zu berücksichtigen, können Entscheidungen transparenter, begründbarer und auch intern leichter vermittelbar gemacht werden. Der in dieser Arbeit entwickelte Bewertungsansatz kann dabei nicht nur auf Testautomatisierung, sondern prinzipiell auch auf andere Toolauswahlentscheidungen, wie zum Beispiel bei Monitoring-Tools übertragen werden. Auch für Unternehmen ohne tiefgehende methodische Vorkenntnisse bietet die MCDA-Matrix aufgrund ihrer klaren Struktur und Flexibilität einen gut nutzbaren Rahmen.

Gleichzeitig zeigt die Arbeit auch, dass solche Bewertungsmodelle kontinuierlich weiterentwickelt und angepasst werden sollten. Im vorliegenden Fall liegt der Fokus auf sechs zentrale Kriterien, um eine praktikable Bewertung zu ermöglichen. In anderen Anwendungsszenarien können zusätzliche Kriterien wie Skalierbarkeit, Datenschutz, Lizenzmodell oder Community-Support eine wichtige Rolle spielen. Auch kann eine noch differenziertere Gewichtung, wie durch Methoden wie den Analytic Hierarchy Process (AHP) oder Gruppenentscheidungen, zusätzliche Erkenntnisse liefern. Die Integration solcher erweiterten Bewertungsverfahren wäre ein sinnvoller Ansatzpunkt für die zukünftige Forschung.
Darüber hinaus wäre es im Rahmen weiterführender Studien interessant, zu untersuchen,

wie sich die Einschätzungen in einer MCDA-Matrix im Zeitverlauf verändern, wenn ein Unternehmen mit einem ausgewählten Tool länger arbeitet und neue Erfahrungen sammelt. Auch der Vergleich zwischen subjektiver Erwartung und tatsächlicher Nutzungserfahrung eines Tools könnte aufschlussreiche Hinweise für die Validität solcher Entscheidungsmodelle liefern. Ein weiterer lohnender Forschungspfad liegt in der Kombination von MCDA mit empirischen Daten aus der Softwareentwicklung. So könnten beispielsweise reale Fehlererkennungsraten oder Metriken aus CI/CD-Pipelines als objektive Ergänzungen zu subjektiven Einschätzungen herangezogen werden, um die Robustheit des Modells zu erhöhen. Insgesamt lässt sich festhalten, dass das in dieser Arbeit entwickelte Modell sowohl praktikabel als auch ausbaufähig ist. Es bietet eine methodisch fundierte Grundlage für praxisnahe Entscheidungen und kann als Ausgangspunkt für eine weitergehende Auseinandersetzung mit systematischer Toolbewertung im Softwarekontext dienen.

Hochschule Neu-Ulm
University of Applied Sciences

Literaturverzeichnis

Ahmed Ramadan, H. Y. B. P. (2024). The Role of Artificial Intelligence and Machine Learning in Software Testing.

Arcuri, A. (2021). Software Testing and Artificial Intelligence: A Survey. *Journal of Systems and Software*.

B. Garcia F. Gortazar, M. G., & Jim´enez, E. (2017). User Impersonation as a Service in End-to-End Testing. *Science and Technology Publications,*

.

Belton, V., & Stewart, T. J. (2002). Multiple Criteria Decision Analysis.

Bichachi, R. (2025). Total Cost of Ownership Explained. *Net Suite*. https://www.netsuite.com/portal/resource/articles/accounting/total-cost-ownership-tco.shtml?utm_source=chatgpt.com

Choudhary, J. P. P. N. (2023). A Systematic Review on Anomaly Detection. *International Journal of Advanced Research in Science Communication and Technology*.

Demetrio, R. (2024). KI-Statistiken: 500+ Fakten, die globale Innovation *bureauworks*.

DocBox. (2025a). https://www.docbox.eu/docbox-inhouse.html

DocBox. (2025b). *Richtlinien*. https://www.docbox.eu/service-kontakt/glossar-der-docbox/dokumentenmanagement.html

Fowler, M. (2018). The Practical Test Pyramid. Arcuri, A. (2021). Software Testing and Artificial Intelligence: A Survey. Journal of Systems and Software.

Garousi, V., Felderer, M., & Hacaloğlu. (2022). A survey of software test automation practices: Benefits, challenges, and best practices. Software Testing. *Verification & Reliability*.

Geldermann, P. D. J., & Lerche, N. (2014). Leitfaden zur Anwendung von Methoden der multikriteriellen Entscheidungsunterstützung

Gregory, J., & Crispin, L. (2015). More Agile Testing: Learning Journeys for the Whole Team. *Addison-Wesley Signature Series*.

Gulp. (2025). *Randstad Professional*. https://www.gulp.de/gulp2/g/spezialisten?hourlyRateCalculator=true&lstvAndOr=und&resultsample=15&mode=content&query=java%20script%20Junior-Entwickler&page=1

ISTQB. (2018). Certified Tester Foundation Level Syllabus Version 2018. International Software Testing Qualifications Board.

ISTQB. (2025). *End-to-End-Testing*. International Software Testing Qualifications Board.

Kampffmeyer, U., & Merkel, B. (1999). DokumentenManagement: Grundlagen und Zukunft. *Project Consult GmbH*.

Mabl. (2025). *Intelligent Test Automation for CI/CD*. https://www.mabl.com/

Market Guide for Content Services Platforms. (2022). *Gartner*.

Martensson, T. (2022). Efficient and effective exploratory testing of large-scale software systems. *Journal of Systems and Software*.

Microsoft. (2025). *Playwright Documentation*. Microsoft. https://playwright.dev/

npm-stat.com. (2025). Playwright Download. *npm-stat.com*.

O'Callaghan, R., & Smits, M. (2005). A Strategy Development Process for Enterprise Content Management.

Ostheimer, B., & Janz, W. (2005). Dokumenten-Management-Systeme – Abgrenzung, Wirtschaftlichkeit, rechtliche Aspekte

Peffers, K., Rothenberger, M., Tuunanen, T., & Vaezi, R. (2007). A design science research methodology for information systems research.

Qutera. (2025). Überblick zum begehrten Testautomatisierungs Tool Playwright - Open Source End-To-End Tests. *Qutera*.

R. Black, E. v. V., and D. Graham. (2012). Foundations of Software Testing: ISTQB Certification. *Andover: Cengage Learning*.

Rammer, D. C. (2024). *KI-Einsatz in Unternehmen in Deutschland*.

Riggert, W. (2019). *Dokumentenmanagement*.

Rwemalika, R., Kintis, M., Papadakis, M., & Traon, Y. L. (2018). Can we automate away the main challenges of end-to-end testing?

Sampath, S., & Sprenkle, S. (2016). "Chapter Four - Advances in Web Application Testing, 2010–2014. *Advances in Computers*.

Sandler, C., & Myers, G. J. (2015). The Art of Software Testing. *WILEY*.

Schär, S. (2018). State-of-the-Art dynamischer Methoden zur multikriteriellen Entscheidungsunterstützung. *Junior Management Science*.

SmartBear. (2025). *How to Perform End-to-End Testing*. SmartBear. https://smartbear.com/learn/automated-testing/what-is-end-to-end-testing/

The State of Intelligent Information Management. Association for Intelligent Information Management. (2023).

Zhang, M., Harman, M., Ma, L., & Liu, Y. . (2020). Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Software Engineering*.