THM | CAMPUS GIESSEN | MNI
Mathematik, Naturwissenschaften und Informatik
TECHNISCHE HOCHSCHULE MITTELHESSEN

Bachelorarbeit

„Smarte Einführung von Prozessautomatisierung in ERP-Systemen: Analyse des Implementierungsaufwands unter Einbeziehung von Künstlicher Intelligenz“

Patrick Mühlhausen 5312094*

01. Juli 2026

Betreut von:
Prof. Dr. Frank Kammer
Jonas Wölfer M.Sc

*FB MNI Mathematischen, Naturwissenschaften, Informatik

Erklärung zur Verwendung von generativer KI

In Übereinstimmung mit den Empfehlungen der Deutschen Forschungsgemeinschaft (DFG) und denen der Zeitschrift *Theoretical Computer Science* erkläre ich hiermit den Einsatz von generativer KI.

Bei der Vorbereitung dieser Arbeit habe ich ChatGPT 45 verwendet, um ausschließlich die Lesbarkeit und Sprache zu verbessern. Nach der Verwendung von ChatGPT 5 habe ich den Inhalt überprüft und nach Bedarf bearbeitet und übernehme die volle Verantwortung für den Inhalt dieser Arbeit.

Gender-Disclaimer

In dieser Arbeit wird aus Gründen der besseren Lesbarkeit das generische Maskulinum verwendet. Weibliche und anderweitige Geschlechteridentitäten werden dabei ausdrücklich mitgemeint, soweit es für die Aussage erforderlich ist.

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Gießen, den 01. Juli 2026 Patrick Mühlhausen

Abstract

Die Automatisierung der Rechnungsverarbeitung mit Robotic Process Automation (RPA) verspricht Effizienzgewinne, scheitert in der Praxis jedoch selten am fehlerfreien Regelfall, sondern an der Behandlung von Ausnahmen. Der Aufwand für dieses Exception-Handling bestimmt maßgeblich die Wirtschaftlichkeit solcher Projekte. Die vorliegende Arbeit untersucht vergleichend, wie sich der Implementierungsaufwand für Exception-Handling-Mechanismen auf drei RPA-Plattformen unterscheidet. Dazu wurde derselbe KI-gestützte Rechnungsverarbeitungsprozess auf UiPath, Microsoft Power Automate und Sema4.ai umgesetzt und in das Enterprise Resource Planning (ERP)-System Dolibarr integriert. Ein gemeinsamer Satz von 30 Rechnungen durchlief jede Lösung; eine ordnerbasierte Ablage diente als Messinstrument für die Straight-Through-, die Exception- und die Bot-Success-Rate sowie die Durchlaufzeit. Ergänzend wurden Wirtschaftlichkeitskennzahlen geschätzt, ein qualitativer Plattformvergleich durchgeführt und einzelne Rechnungen fallweise betrachtet.

Die Ergebnisse zeigen, dass sich die Plattformen weniger im eigentlichen Exception-Handling-Mechanismus unterscheiden als in der Qualität ihrer KI-Extraktion und der Ergonomie ihrer Umsetzung. Diese Faktoren bestimmen die Häufigkeit manueller Eingriffe und damit die Wirtschaftlichkeit. UiPath extrahierte am zuverlässigsten und arbeitete am autonomsten, während Power Automate die meisten Extraktionsfehler und den höchsten Umsetzungsaufwand aufwies; Sema4.ai war transparent und schnell umzusetzen, litt aber unter unsicherer Extraktion. Entgegen der verbreiteten Annahme war die Low-Code-Lösung nicht die schnellste. Ab einem mittleren Rechnungsvolumen amortisiert sich die Automatisierung innerhalb weniger Monate. Entscheidend für die Wirtschaftlichkeit ist damit weniger die Plattformwahl als die Zuverlässigkeit der Extraktion und die Kalibrierung der Konfidenz.

Inhaltsverzeichnis

1. Einleitung **1**
- 1.1. Problemstellung und Relevanz . . . 1
- 1.2. Zielsetzung der Arbeit . . . 1
- 1.3. Aufbau der Arbeit . . . 2
- 1.4. Abgrenzung der Arbeit . . . 2

2. Theoretische Grundlagen **3**
- 2.1. Enterprise Resource Planning-Systeme . . . 3
- 2.2. Robotic Process Automation . . . 4
- 2.3. Künstliche Intelligenz gestützte Dokumentenverarbeitung . . . 4
- 2.4. Dokumentenextraktion und Confidence Scores . . . 5
- 2.5. Metriken . . . 5
  - 2.5.1. Grundlagen des Exception Handlings in der Rechnungsprozessautomatisierung . . . 5
  - 2.5.2. Wirtschaftlichkeitsmetriken . . . 7
  - 2.5.3. Technische Metriken . . . 8
- 2.6. Rechnungsverarbeitung . . . 9

3. Ausgewählte Technologien **10**
- 3.1. Dolibarr ERP (Document Understanding) . . . 10
- 3.2. UiPath . . . 11
- 3.3. Microsoft Power Automate . . . 12
- 3.4. Sema4.ai (Nanonets) . . . 13

4. Verwandte Arbeiten **14**
- 4.1. Automatisierung der Rechnungsverarbeitung mit RPA . . . 14
- 4.2. Intelligente Dokumentenverarbeitung und KI-Erweiterung . . . 14
- 4.3. Vergleich von RPA-Plattformen . . . 14
- 4.4. Einordnung der eigenen Arbeit . . . 15

5. Methodik **16**
- 5.1. Vorstellung des Anwendungsszenarios . . . 16
  - 5.1.1. Fachlicher Rahmen und eingesetzte Systeme . . . 16
  - 5.1.2. Zu extrahierende Datenfelder . . . 16
  - 5.1.3. Statusbasierte Ablagesteuerung . . . 17
- 5.2. Auswahl der RPA-Plattformen . . . 17
  - 5.2.1. Auswahlkriterien . . . 17
  - 5.2.2. Gegenüberstellung . . . 18
- 5.3. Prozessaufbau . . . 18
- 5.4. Auswahl der Metriken . . . 23
  - 5.4.1. Wirtschaftlichkeitsmetriken . . . 23
  - 5.4.2. Prozesseffizienzmetriken und HITL-Validierung . . . 23
  - 5.4.3. Metriken zum Imagegewinn . . . 24
  - 5.4.4. Qualitativer Plattformvergleich . . . 25
  - 5.4.5. Übersicht der ausgewählten Metriken . . . 25
- 5.5. Testaufbau . . . 26
  - 5.5.1. Testumgebung und Systemkonfiguration . . . 26
  - 5.5.2. Struktur des Testkorpus und vordefinierte Fehlerszenarien . . . 26
  - 5.5.3. Versuchsdurchführung und Datenerhebung . . . 27
  - 5.5.4. Messung und Zuordnung der Evaluationsmetriken . . . 27

6. Implementierung der Prototypen **29**
6.1. Gemeinsame Anforderungen an alle Prototypen . . . 29
6.2. Implementierung mit UiPath und Document Understanding . . . 29
6.2.1. Standardprozess . . . 29
6.2.2. Umsetzung des Exception-Handlings . . . 31
6.2.3. Umsetzung der manuellen Validierung . . . 32
6.2.4. Logging und Fehlerablage . . . 33
6.3. Implementierung mit Microsoft Power Automate und AI Builder . . . 33
6.3.1. Umsetzung des Exception-Handlings . . . 35
6.3.2. Umsetzung der manuellen Validierung . . . 35
6.3.3. Logging und Fehlerablage . . . 37
6.4. Implementierung mit Sema4.ai und Nanonets . . . 37
6.4.1. Standardprozess . . . 37
6.4.2. Umsetzung des Exception-Handlings . . . 39
6.4.3. Umsetzung der manuellen Validierung . . . 39
6.4.4. Logging und Fehlerablage . . . 40

7. Ergebnisse **41**
7.1. Zielablage der Rechnungen . . . 41
7.2. Prozesseffizienz . . . 41
7.3. Durchlaufzeit . . . 42
7.4. Verhalten in den Ausnahmefällen . . . 42
7.4.1. Geschäftsfehler: Betragsabweichung . . . 42
7.4.2. Systemfehler und Wiederanlauf . . . 42
7.4.3. Nicht prüfbare Fehlerklassen . . . 43
7.5. Implementierungsaufwand . . . 43
7.6. Qualitativer Plattformvergleich . . . 43
7.7. Wirtschaftliche Kennzahlen . . . 44
7.8. On-Time Payment Rate (Proxy) . . . 45
7.9. Zusammenfassung der Messergebnisse . . . 45

8. Diskussion **46**
8.1. Extraktionsqualität als Haupttreiber der Ergebnisse . . . 46
8.2. STP-Rate und tatsächliche Autonomie . . . 47
8.3. Implementierungsaufwand: Low-Code ist nicht automatisch schneller . . . 47
8.4. Wirtschaftlichkeit: Hat sich die Automatisierung gelohnt? . . . 48
8.5. Imagegewinn durch schnellere Verarbeitung . . . 48
8.6. Fallbasierte Betrachtung ausgewählter Rechnungen . . . 49
8.6.1. Standardfall: eine korrekte Rechnung . . . 49
8.6.2. Geschäftsfehler: falscher Bruttobetrag . . . 50
8.6.3. Fehlextraktion im Anzahl-Feld: „21 km á 0,45 €“ . . . 50
8.6.4. Fälligkeitsdatum in Textform: „In 10 Tagen bitte zahlen“ . . . 51
8.7. Einordnung in die Literatur . . . 52
8.8. Herausforderungen bei der Umsetzung . . . 53
8.9. Beantwortung der Forschungsfragen . . . 55
8.10. Limitationen . . . 56

9. Fazit **57**

10. Ausblick **58**

Abbildungsverzeichnis **59**

Tabellenverzeichnis **60**

Listingsverzeichnis **61**

Literaturverzeichnis **62**

A. Anhang **66**

B. Anhang **72**

1. Einleitung

1.1. Problemstellung und Relevanz

Die Automatisierung von Geschäftsprozessen hat sich in der Unternehmenspraxis bereits fest etabliert, da sie bewiesenermaßen Effizienzsteigerungen, Fehlerreduktion und eine Entlastung personeller Ressourcen verspricht [1, S. 1, 3–4]. Insbesondere Enterprise Resource Planning (ERP)-Systeme bieten hierfür ein relevantes Anwendungsfeld. Sie integrieren verschiedene betriebliche Funktionen und Daten in ein zentrales Informationssystem [2, S. 4]. Als operative Plattform bilden sie das Rückgrat der betrieblichen Informationsverarbeitung [2, S. 10]. Sollen jedoch komplexe Prozesse automatisiert werden, weisen bestehende ERP-Systeme im laufenden Betrieb oft eine ungenügende Anpassungsfähigkeit auf und stoßen als starre Systeme an ihre Grenzen [2, S. 50]. Zudem stellt insbesondere die Verarbeitung unstrukturierter Daten ein Hindernis dar, was den ergänzenden Einsatz von Künstlicher Intelligenz (KI) und Machine Learning zur Datenextraktion erfordert [1, S. 9-10].

Technologien wie Robotic Process Automation (RPA) ermöglichen es, systemübergreifende Prozessschritte flexibel zu automatisieren, indem Software-Bots menschliche Interaktionen an der Benutzeroberfläche nachahmen und so fehlende Programmierschnittstellen (APIs) zu Altsystemen überbrücken [1, S. 1, 4]. Das zentrale Problem bei der praktischen Umsetzung ist jedoch nicht der fehlerfreie Regelfall, der sogenannte „Happy Path“, sondern das systematische Auftreten von Abweichungen [1, S. 5, 9]. Die Implementierung von stabilen Mechanismen zur Ausnahmebehandlung (Exception-Handling) bestimmt maßgeblich den technischen und zeitlichen Gesamtaufwand eines Projekts, da der Versuch, Ausnahmen abseits des „Happy Paths“ zu automatisieren, die Entwicklungszeit schnell verdoppeln kann [1, S. 5].

Unternehmen stehen vor der Herausforderung, dass Bots oft nicht mit ausreichenden Anweisungen programmiert sind, um alle auftretenden Abweichungen, etwa durch geänderte Benutzeroberflächen oder angepasste Geschäftsregeln, eigenständig zu bewältigen. Da ein ineffizientes Exception-Handling dazu führt, dass Bots falsche Pfade einschlagen oder blockieren und manuelles Eingreifen erfordern, können solche Fehlerpfade die angestrebten Vorteile der Automatisierung und somit die Rentabilität, Return on Investment (ROI) zunichtemachen. Bisher fehlt es in der wissenschaftlichen Literatur jedoch an fundierten Untersuchungen zur technischen Implementierung von RPA, da die Bot-Entwicklung in der Praxis noch einen stark manuellen, fehleranfälligen Aufwand darstellt und neuartige Frameworks für das Exception-Handling noch weitgehend fehlen [1, S. 10-12].

1.2. Zielsetzung der Arbeit

Ziel dieser Arbeit ist es, die smarte Einführung von Prozessautomatisierung in ERP-Systemen exemplarisch am Beispiel eines KI-gestützten Rechnungsverarbeitungsprozesses zu untersuchen. Im Fokus steht die vergleichende Analyse des zeitlichen und technischen Implementierungsaufwands von Exception-Handling-Mechanismen auf ausgewählten RPA-Plattformen. Daraus ergeben sich die folgenden Forschungsfragen:

1. Wie unterscheiden sich ausgewählte RPA-Plattformen in Bezug auf die Umsetzung von Exception-Handling-Mechanismen in KI-gestützten, ERP-integrierten Rechnungsverarbeitungsprozessen hinsichtlich ihrer Wirtschaftlichkeit?
    - 1.1. Welche Implementierungsschritte sind für die Automatisierung des gewählten Rechnungsverarbeitungsprozesses erforderlich?
    - 1.2. Welche Arten von Exceptions (System- und Geschäftsfehlern) treten bei der automatisierten, KI-gestützten Rechnungsverarbeitung auf?
    - 1.3. Wie unterscheiden sich die ausgewählten RPA-Plattformen beim Implementierungsaufwand für diese Exception-Handling-Mechanismen?

1.4. Wie stellt sich die Wirtschaftlichkeit der betrachteten RPA-Lösungen anhand von Metriken wie dem Return on Investment, der Amortisationsdauer und der quantitativen FTE-Einsparung (Full-Time Equivalent) dar?

1.5. Wie gut lässt sich ein unsicher oder falsch extrahiertes Attribut validieren, korrigieren und anschließend in den automatisierten ERP-Prozess zurückführen?

1.3. Aufbau der Arbeit

Die vorliegende Arbeit ist in mehrere aufeinander aufbauende Abschnitte gegliedert, um eine strukturierte und methodisch nachvollziehbare Beantwortung der Forschungsfragen zu gewährleisten.

Nach der Einleitung werden zunächst die theoretischen Grundlagen erarbeitet. Neben der Einordnung von ERP-Systemen liegt der Fokus hierbei auf der robotergestützten Prozessautomatisierung, der KI-gestützten Dokumentenverarbeitung sowie der Klassifikation von Ausnahmefehlern und der Definition technischer und wirtschaftlicher Bewertungsmetriken. Dieser Abschnitt schließt mit der Betrachtung der prozessualen Rahmenbedingungen der Rechnungsverarbeitung ab. Darauf aufbauend werden die ausgewählten Technologien in ihrem technischen Kontext vorgestellt. Eine Aufarbeitung verwandter Arbeiten rundet das theoretische Fundament ab.

Der empirische Teil der Arbeit beginnt mit der Methodik, welche das Forschungsdesign umfassend beschreibt. Dies beinhaltet die Definition des konkreten Anwendungsszenarios, die Auswahlkriterien für die Plattformen, den exakten Prozessaufbau, die Auswahl der Metriken sowie den Testaufbau. Im Zuge der anschließenden Realisierung wird die praktische Implementierung der Prototypen dokumentiert. Nach der Festlegung gemeinsamer Anforderungen wird die schrittweise Umsetzung in den drei RPA-Plattformen dargelegt, wobei pro Plattform der Standardprozess, das Exception-Handling, die manuelle Validierung sowie das Logging und die Fehlerablage im Vordergrund stehen.

In den anschließenden Ergebnissen werden die quantitativen und qualitativen Daten der durchgeführten Testreihen aufbereitet und im Hinblick auf die Prozesseffizienz, Durchlaufzeiten, das Verhalten in Ausnahmefällen sowie den tatsächlichen Implementierungsaufwand vergleichend gegenübergestellt. Der darauffolgende Abschnitt widmet sich der Diskussion dieser Erkenntnisse vor dem Hintergrund der Forschungsfragen. Die Arbeit schließt mit einem Fazit, das die wesentlichen Ergebnisse zusammenfasst und einen Ausblick auf zukünftige technologische Entwicklungen gibt.

1.4. Abgrenzung der Arbeit

Die vorliegende Arbeit betrachtet einen klar umrissenen Ausschnitt der Rechnungsverarbeitung. Untersucht wird ausschließlich die Verarbeitung eingehender Lieferantenrechnungen; die Erstellung eigener Ausgangsrechnungen bleibt außen vor. Als ERP-System dient allein Dolibarr, andere Systeme werden nicht betrachtet. Die Rechnungen liegen als unstrukturierte PDF-Dateien vor; strukturierte E-Rechnungsformate werden nicht behandelt. Der Rechnungseingang erfolgt über einen Dateiordner und nicht über ein E-Mail-Postfach. Die Konfidenzschwelle für die Human-in-the-Loop-Validierung ist auf 80 % festgelegt und wird nicht optimiert.

Sicherheitsaspekte sind nicht Gegenstand der Untersuchung. Fragen der IT-Sicherheit – etwa die Absicherung gegen manipulierte Rechnungen (Prompt Injection), der Umgang mit Zugangsdaten und Programmierschnittstellen-Schlüsseln, die rechtekonforme Ausführung der Bots nach dem Least-Privilege-Prinzip sowie datenschutzrechtliche Anforderungen bei der Übermittlung an externe Cloud-Extraktionsdienste – bleiben ausgeklammert. Ebenso werden die Revisionssicherheit der automatischen Buchungen und weitergehende Maßnahmen zur Betrugserkennung nicht betrachtet.

2. Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen definiert, die für das Verständnis und die anschließende Evaluation der automatisierten Rechnungsprüfung notwendig sind.

Zunächst werden Enterprise Resource Planning (ERP)-Systeme eingeführt (Abschnitt 2.1). Darauf folgt die theoretische Einordnung von Robotic Process Automation (Abschnitt 2.2). Die Abschnitte 2.3 und 2.4 beleuchten die künstliche intelligenzgestützte Dokumentenverarbeitung sowie die zugehörige Dokumentenextraktion und die Funktionsweise von Confidence Scores. Abschnitt 2.5 etabliert anschließend das Metrikmodell, welches sich in das softwaretechnische Exception Handling, die Wirtschaftlichkeitsmetriken sowie die technischen Leistungskennzahlen untergliedert. Abschnitt 2.6 legt schließlich die prozessualen und rechtlichen Rahmenbedingungen der Rechnungsverarbeitung dar.

2.1. Enterprise Resource Planning-Systeme

Ein Enterprise Resource Planning-System beschreibt ein modular aufgebautes Paket für betriebswirtschaftliche Standardanwendungssoftware, das alle geschäftsrelevanten Ressourcen wie Material, Personal, Kapazitäten und Finanzen unternehmensweit plant, steuert und verwaltet [3, S. 120]. Diese Systeme fungieren als zentrale operative Plattform und bilden das Rückgrat der betrieblichen Informationsverarbeitung, indem sie die Schnittstelle zwischen der technologischen Infrastruktur und den operativen Fachprozessen bilden, um Informationen als kritische Erfolgsressource zielgerichtet zu nutzen [2, S. 10].

Das wesentliche Abgrenzungskriterium zu historischen Insel- oder Legacy-Lösungen liegt im Paradigma der datentechnischen und prozessualen Integration [2, S. 7]. ERP-Systeme operieren auf einem einheitlichen Datenmodell und einer zentralen Datenbank, wobei jede transaktionale Änderung zu einer Echtzeit-Aktualisierung der Datenbasis führt, wodurch konsistente Informationen unmittelbar für alle angrenzenden funktionalen Module wie das Finanzwesen, das Controlling oder die Produktion bereitstehen [3, S. 119-120].

Die Entstehung moderner ERP-Architekturen ist das Ergebnis einer kontinuierlichen Evolution betrieblicher Planungskonzepte seit den 1970er-Jahren. Diese Entwicklung verlief sequentiell von rein materialwirtschaftlich orientierten Material Requirement Planning Systemen (MRP) über die Integration von Kapazitätsrückkopplungen zur ganzheitlichen Manufacturing Resource Planning Produktionsressourcenplanung (MRP II) in den 1980er-Jahren. In den 1990er-Jahren etablierten sich schlussendlich die branchenunabhängigen, unternehmensweiten ERP-Systeme, welche zur Jahrtausendwende durch internetba-sierte Schnittstellen und Interoperabilität für globale Netzwerke und Supply Chains erweitert wurden. [2, S. 8-9, 28]

Aus strategischer Sicht ist die Implementierung eines solchen Systems kein reines Informationstechnologie-Projekt (IT-Projekt), sondern ein tiefgreifender organisatorischer Transformationsprozess. ERP-Systeme erzwingen durch ihre prozessuale Ausrichtung die Abkehr von funktionalen Abteilungssilos hin zu integrierten Workflows und Geschäftsprozessen. Dabei entsteht in der Praxis ein zentraler Konflikt zwischen Standardisierung und strategischer Differenzierung: Die Nutzung vordefinierter Standardprozesse verspricht signifikante Kosten- und Zeitvorteile durch den Abbau von Schnittstellen, birgt jedoch das Risiko, unternehmensspezifische Wettbewerbsvorteile zu verlieren. [3, S. 117-119]

Werden die organisatorischen Abläufe zwingend an die Vorgaben der Standardsoftware angepasst, kann dies die organisatorische Flexibilität und Anpassungsfähigkeit gegenüber dynamischen Marktveränderungen empfindlich einschränken [2, S. 14, 50].

2.2. Robotic Process Automation

Robotic Process Automation beschreibt eine softwarebasierte Technologie zur Automatisierung strukturierter, repetitiver und regelbasierter Geschäftsprozesse durch den Einsatz von spezialisierten Software-Agenten, sogenannten Software-Bots [3, S. 395]; [1, S. 1]. Im Gegensatz zu traditionellen Ansätzen der Systemintegration, die tiefgreifende Modifikationen der Anwendungslogik oder die Entwicklung dedizierter APIs erfordern, operiert RPA primär auf der User Interface Präsentationsschicht (UI) bestehender Softwaresysteme [3, S. 396–397]; [1, S. 1, 8]. Die Bots imitieren hierbei die menschliche Interaktion mit der Benutzeroberfläche, indem sie Tastatureingaben simulieren, Mausklicks ausführen, Bildschirminhalte auslesen und Daten zwischen isolierten Applikationen transferieren, ohne dass die zugrunde liegende IT-Infrastruktur verändert werden muss [3, S. 396]; [1, S. 1, 5, 8–9].

In der wissenschaftlichen Literatur wird die technologische Architektur von Enterprise-RPA-Systemen typischerweise in eine dreigeteilte Struktur unterteilt, die sich aus der Entwicklungsumgebung, der Orchestrierungskomponente und den Ausführungseinheiten zusammensetzt [1, S. 8]. Die Entwicklungsumgebung dient der Modellierung der Prozesslogik, die entweder visuell über Flussdiagramme und Sequenzen (Low-Code) oder textbasiert über Skripte erfolgt [3, S. 395]; [1, S. 8–9]. Die Orchestrierungskomponente fungiert als zentrales Kontrollzentrum, welches für die Bereitstellung, die Überwachung, das Logging, die Lastverteilung und die Terminierung der Bots verantwortlich ist [1, S. 8, 10]. Die Ausführungseinheiten, die eigentlichen Bots, werden prozessual in attendierte und unattendierte Roboter differenziert [3, S. 396]; [1, S. 8]. Während attendierte Bots im Vordergrund auf Benutzerstationen ausgeführt werden und eine direkte menschliche Interaktion erfordern, agieren unattendierte Bots vollständig autonom auf virtuellen Serverumgebungen [3, S. 396]; [1, S. 8] und eignen sich primär für datenintensive Hintergrundprozesse wie die allgemeine Transaktionsverarbeitung oder das Auslesen von E-Mail-Anhängen [3, S. 396]; [1, S. 1, 3].

2.3. Künstliche Intelligenz gestützte Dokumentenverarbeitung

Die Integration von KI in die Dokumentenverarbeitung, in der Fachliteratur als Intelligent Document Processing (IDP) bezeichnet, markiert einen Paradigmenwechsel gegenüber klassischen Automatisierungsansätzen. Traditionelle, regelbasierte RPA-Bots stoßen bei der Verarbeitung von Eingangsrechnungen an enge Grenzen, da diese Belege als semistrukturierte oder unstrukturierte Datenquellen vorliegen und keinem einheitlichen geometrischen Layout folgen. [4, S. 2]

IDP-Systeme lösen diese Restriktion auf, indem sie Verfahren des maschinellen Lernens und der tiefen neuronalen Netze einsetzen, um Dokumente nicht nur zeichenbasiert auszulesen, sondern deren logischen Kontext zu interpretieren. Technologisch basiert diese intelligente Verarbeitung auf einer hybriden Architektur aus Computer Vision und Natural Language Processing (NLP). [5]

Während Convolutional Neural Networks (CNNs) für die visuelle Struktur- und Segmentierungsanalyse eingesetzt werden – beispielsweise zur Identifikation von Tabellenstrukturen, Rechnungspositionen oder Firmenlogos [4, S. 3] –, übernehmen Transformer-basierte Sprachmodelle die semantische Textextraktion [5]. Das System liest somit Werte wie das Rechnungsdatum oder den Bruttobetrag nicht mehr über feste Pixelkoordinaten oder Zonen aus, sondern identifiziert sie anhand sprachlicher Muster und syntaktischer Beziehungen im Textgefüge [4, S. 2]. Dies reduziert den initialen Schablonen-Konfigurationsaufwand drastisch, führt jedoch beim Übergang von regelbasierter RPA zu intelligenter Automatisierung zu einer probabilistischen statt deterministischen Arbeitsweise [6, S. 24], was die mathematische Überwachung der Extraktionsgüte durch Confidence Scores zwingend erforderlich macht [4, S. 2, 4].

2.4. Dokumentenextraktion und Confidence Scores

Der Prozess der Dokumentenextraktion nutzt Technologien der KI und des maschinellen Lernens, um unstrukturierte visuelle Informationseinheiten in ein strukturiertes Format zu transformieren [6, S. 13]. Dieses strukturierte Format, wie beispielsweise JSON, kann anschließend vom RPA-Bot an das ERP-System oder eine Datenbank übergeben werden [4, S. 4] [7, S. 3].

Da dieser Extraktionsprozess auf statistischen Algorithmen und Mustererkennung beruht, liefert die KI-Komponente für jedes extrahierte Datenfeld einen sogenannten Konfidenzwert (Confidence Score). Dieser mathematische Wert drückt aus, mit welcher prozentualen Wahrscheinlichkeit das KI-Modell davon ausgeht, dass das extrahierte Zeichen oder Wort der tatsächlichen Information auf dem Beleg entspricht und dient als Grundlage für nachgelagerte Validierungen. [4, S. 4]

Für das automatisierte Exception-Handling ist der Confidence Score eine zentrale Steuerungsgröße auf Systemebene. In den Workflows der RPA-Plattformen werden feste Schwellenwerte (Confidence Thresholds) definiert, die als logische Weichen fungieren und die Balance zwischen automatischer Freigabe (Auto-Approval) und manueller Überprüfung steuern. Erreicht eine Rechnung auf allen kritischen Feldern, wie Rechnungsnummer, Gesamtbetrag und Umsatzsteuer-Identifikationsnummer (ID), einen Konfidenzwert, der über dem Schwellenwert liegt, wird der Datensatz als sicher eingestuft und für die Dunkelbuchung, das sogenannte Straight-Through Processing (STP), freigegeben [4, S. 2] [6, S. 7]. Unterschreitet hingegen auch nur ein einziger Pflichtwert diese Grenze, wird die automatische Verarbeitung gestoppt und der Vorgang in eine manuelle Ausnahmebehandlung HITL-Workflow (Human-in-the-Loop-Workflow) überführt [4, S. 4].

2.5. Metriken

In diesem Kapitel werden die Metriken definiert, um eine objektive und neutrale Vergleichsbasis für die drei RPA-Implementierungen zu schaffen. Dazu wird zunächst das Exception Handling hergeleitet und die technischen Leistungskennzahlen wie die STP-, Exception- und Bot-Success-Rate definiert. Abschließend werden die Wirtschaftlichkeitsmetriken etabliert, welche diese operativen Betriebsdaten in finanzielle Kennzahlen übersetzen.

2.5.1. Grundlagen des Exception Handlings in der Rechnungsprozessautomatisierung

Das Exception Handling, beziehungsweise die strukturierte Ausnahmebehandlung, beschreibt die softwaretechnische Architektur zur Erkennung, Protokollierung und Behebung von Abweichungen vom standardmäßigen Prozesspfad innerhalb eines automatisierten Workflows [1, S. 11-12]. In der Rechnungsprozessautomatisierung ist die Einführung einer robusten Ausnahmebehandlung eine zwingende Voraussetzung für den stabilen Betrieb, da die Kombination aus unstrukturierten Eingangsdaten und verteilten Systemlandschaften eine inhärente Fehleranfälligkeit besitzt [4, S. 1-2]. Ein unvollständiges Kontrolldesign führt dazu, dass nicht abgefangene Ausnahmen den Software-Bot blockieren oder im schlimmsten Fall zu fehlerhaften Buchungssätzen im nachgelagerten ERP-System führen [4, S. 2], was den manuellen Korrekturaufwand drastisch erhöht und die Wirtschaftlichkeit des Gesamtprojekts mindert [1, S. 11-12].

Für den Aufbau stabiler Automatisierungsworkflows etabliert die Praxis hierzu eine strikte Taxonomie zwischen zwei fundamentalen Fehlerklassen, die grundlegend differenzierte technologische Behandlungsstrategien erfordern: Anwendungsfehler (Application Exceptions) und Geschäftsfehler (Business Exceptions) [8]. Während Anwendungsfehler primär technische Probleme wie blockierende oder nicht reagierende Applikationen betreffen und durch automatisierte Wiederholungsroutinen (Retrys) der RPA-Plattform abgefangen werden können, resultieren Geschäftsfehler aus der Verletzung prozessualer Vorgaben durch unvollständige oder inkorrekte Daten [8] [1, S. 11-12]. Solche logischen Geschäftsfehler

verlangen zwingend eine inhaltliche manuelle Validierung, auch Human-in-the-Loop (HITL) genannt, da ein automatisierter Neustart der Transaktion das grundlegende inhaltliche Datenproblem nicht lösen würde [8] [4, S. 6-7].

Fallback und Retry Mechanismen (Systemfehler) Systemfehler (Application Exceptions) sind technischer Natur und werden durch Instabilitäten der zugrunde liegenden IT-Infrastruktur oder der beteiligten Software-Applikationen verursacht [8]. Typische Beispiele im Kontext der Rechnungsverarbeitung umfassen unvorhergesehene Netzwerk-Timeouts, temporär nicht erreichbare Schnittstellen des ERP-Systems, plötzliche Systemabstürze beziehungsweise nicht reagierende ERP-Clients oder unerwartete Änderungen in der grafischen Benutzeroberfläche, die das Screen Scraping des Bots blockieren [1, S. 12]. Ein wesentliches Merkmal von Systemfehlern ist ihre meist temporäre, also nur vorübergehende Natur. Gleichzeitig ist der Beleg in der Regel vollständig korrekt [8].

Da ein Großteil der Systemfehler auf temporären Server- oder Infrastrukturproblemen beruht, implementieren moderne RPA-Architekturen eine zeitlich und quantitativ parametrisierte automatische Wiederholungslogik (Auto-Retry), um die Fehlertoleranz des Gesamtsystems zu erhöhen. Scheitert beispielsweise der Zugriff auf eine angebundene Datenbank, wartet der Bot ein definiertes Intervall und startet einen erneuten Verbindungsversuch, da sich blockierte Anwendungen oftmals von selbst wieder stabilisieren. [8]

Sollten diese autonomen Versuche fehlschlagen, greift das System auf definierte Fallback-Routinen zurück, um eine Kaskadierung des Fehlers auf nachfolgende Prozesstakte zu verhindern. Dies beinhaltet das kontrollierte Zurücksetzen der aktuellen Transaktion im Sinne eines informationstechnischen Rollbacks zur Freigabe blockierter Systemressourcen [1, S. 9], sowie das anschließende Verschieben der betroffenen Rechnung in eine isolierte Fehler-Warteschlange (Dead Letter Channel) [9]. Durch diese prozessuale Isolation in der Ausnahmebehandlung kann der Bot autonom und verzögerungsfrei mit dem nächsten Beleg in der zentralen Warteschlange fortfahren [8] [9], um einen möglichst hohen Grad an Dunkelverarbeitung, dem sogenannten Straight-Through Processing, aufrechtzuerhalten [6, S. 7].

Human-in-the-Loop (Geschäftsfehler) Geschäftsfehler hingegen basieren bei fehlerfrei operierenden technischen Systemen auf der Verletzung von definierten betriebswirtschaftlichen, prozessualen oder gesetzlichen Regeln [8] [1, S. 12]. In der Rechnungsverarbeitung manifestiert sich dies primär durch mathematische Abweichungen beim Drei-Wege-Abgleich zwischen Rechnung, Bestellung und Wareneingang [2, S. 196], das Fehlen steuerrechtlicher Pflichtangaben nach § 14 UStG [10, vgl. Art. 23], unzureichende Confidence Scores der KI-basierten Dokumentenextraktion [4, S. 2].

Im Gegensatz zu Systemfehlern sind Geschäftsfehler deterministisch sowie permanent; eine wiederholte Ausführung unter identischen Bedingungen führt reproduzierbar zum selben Fehler. Da Geschäftsfehler eine intellektuelle Bewertung oder eine manuelle Datenkorrektur erfordern, können sie nicht durch algorithmische Wiederholungen gelöst werden. [8]

Hier greift das theoretische Konzept des Human-in-the-Loop, welches im Rahmen einer sogenannten hybriden RPA-Architektur eine Arbeitsteilung zwischen menschlicher Kognition und maschineller Effizienz etabliert [7, S. 295-300]. Sobald ein Bot einen Geschäftsfehler identifiziert, isoliert das System die betroffene Transaktion, ohne den Gesamtprozess abzubrechen [8]. Die fehlerhaften Datenpunkte werden an eine spezialisierte Interaktionsschnittstelle der RPA-Plattform übergeben, die eine visuelle Validierungsmaske für den zuständigen Sachbearbeiter erzeugt [4, S. 2]. Der Mensch agiert hierbei als Supervisor, korrigiert den fehlerhaften Wert manuell oder erteilt eine bewusste Sonderfreigabe. Nach dieser menschlichen Interaktion übernimmt der Bot die validierten Daten wieder in den Workflow, um die finale Verbuchung im ERP-System durchzuführen [7, S. 18].

2.5.2. Wirtschaftlichkeitsmetriken

Die ökonomische Bewertung von Automatisierungsprojekten im IT-Bereich erfordert den Übergang von rein technischen Leistungskennzahlen zu standardisierten betriebswirtschaftlichen Investitionsrechenverfahren [2, S. 279–280]. Da die Einführung von intelligenten RPA-Lösungen mit initialen Software-Lizenzkosten, Infrastrukturkosten und personellen Entwicklungsaufwänden verbunden ist [2, S. 289–290], muss der finanzielle Netto-Nutzen im Rahmen einer Wirtschaftlichkeitsanalyse transparent dargelegt werden [6, S. 15]. Die Evaluation nutzt dafür mehrdimensionale Metriken, welche die Rentabilität [6, S. 9], die Kosteneffekte [2, S. 280] und die personelle Entlastung der Fachbereiche abbilden [1, S. 3].

Return on Investment Der Return on Investment repräsentiert eine klassische Kennzahl zur Messung der Rendite einer Kapitaleinlage und setzt den Nettogewinn eines Projekts ins Verhältnis zum eingesetzten Gesamtkapital [2, S. 280]. Im Kontext dieser RPA-Untersuchung berechnet sich der ROI aus der Differenz zwischen den aggregierten operativen Kosteneinsparungen durch die Bot-Einführung und den Gesamtinvestitionskosten Capital Expenditure (CapEx), wie Entwicklungskosten und Lizenzgebühren [2, S. 289–290], dividiert durch ebendiese Investitionskosten [6, S. 9]. Da das Exception Handling und die damit verbundenen manuellen Nacharbeitszeiten bei Geschäftsfehlern direkte laufende Betriebskosten Operational Expenditure (OpEx) verursachen [2, S. 279–280], beeinflusst die prozessuale Effizienz der Fehlerbehandlung auf den einzelnen RPA-Plattformen den resultierenden ROI-Prozentsatz maßgeblich [1, S. 11–12].

Amortisationsdauer Die Amortisationsdauer (Payback Period) ist eine dynamische oder statische Kennzahl der Investitionsrechnung, welche den Zeitraum bestimmt, den ein Projekt benötigt, um die initialen Anschaffungs- und Implementierungskosten durch die im laufenden Betrieb erzielten Rückflüsse vollständig wieder einzuspielen [2, S. 280]. In der RPA-Praxis korreliert die Amortisationsdauer stark mit dem initialen Implementierungsaufwand für Sonderfälle: Erfordert eine Plattform aufgrund komplexer fehlertechnischer Modellierungen eine lange Entwicklungsphase, verschiebt sich der Amortisationszeitpunkt nach hinten [3, vgl. Abb. 13.7] [1, S. 11–12]. Eine kurze Amortisationsdauer von empirisch oft wenigen Monaten [3, S. 401] bis hin zu maximal zwei Jahren [2, S. 289] gilt in der Wirtschaftsinformatik als primärer Indikator für eine erfolgreiche, risikoarme Prozessautomatisierung.

FTE-Einsparung Die Kennzahl des Full-Time Equivalent (FTE), im deutschen Sprachraum als Vollzeitäquivalent bezeichnet, dient der Standardisierung und Messung von personellen Arbeitsressourcen. Eine FTE-Einsparung von 1,0 bedeutet hierbei, dass die automatisierte Lösung das Arbeitszeitäquivalent einer Vollzeitkraft im Bereich der manuellen Rechnungsverarbeitung komplett kompensiert [1, S. 3]. In dieser Arbeit wird die FTE-Einsparung über die Differenz der manuellen Bearbeitungszeit vor der Automatisierung und der verbleibenden menschlichen Interaktionszeit im automatisierten Prozess (HILT) berechnet, was die durch RPA erzielte Reduktion der Prozesszykluszeiten quantifiziert [11]. Je effizienter die RPA-Plattform den Sachbearbeiter bei der Fehlerklärung unterstützt und je höher die Dunkelbuchungsquote, beziehungsweise das STP [6, S. 7], ausfällt, desto größer ist die realisierte FTE-Einsparung durch den verringerten manuellen Workload [11]. Diese operativen Zeit- und Ressourceneinsparungen stellen wiederum die wichtigste quantitative Eingangsgröße für die Berechnung des ROI und der Amortisationsdauer dar [11] [2, S. 279–280].

2.5.3. Technische Metriken

Straight-Through Processing Rate Die Straight-Through Processing Rate, im betriebswirtschaftlichen Kontext auch als Dunkelverarbeitungs- oder berührungslose Verarbeitungsquote bezeichnet, gilt als die zentrale Effizienzmetrik durchgehend digitalisierter Geschäftsprozesse [12, vgl. S. 3–5]. Sie definiert den prozentualen Anteil der Transaktionen, die einen Prozess vom initialen Dateneingang bis zum finalen Systemabschluss vollkommen autonom und ohne jeden manuellen Eingriff eines menschlichen Sachbearbeiters durchlaufen [12].

In der KI-gestützten Rechnungsautomatisierung setzt eine hohe STP-Rate voraus, dass sowohl die technologische Infrastruktur (fehlerfreie Systemverfügbarkeit) als auch die mathematischen Modelle der Dokumentenextraktion (fehlerfreie Erkennung über der Konfidenzschwelle) fehlerfrei ineinandergreifen [4, S. 2]. Da jede manuelle Interaktion innerhalb einer Human-in-the-Loop-Schnittstelle die STP-Rate mathematisch auf null setzt, ist diese Metrik der sensitivste Indikator für den tatsächlichen Automatisierungsgrad des operativen Tagesgeschäfts [12, vgl. S. 3–5].

Exception Rate Als mathematisches Gegenstück zur Dunkelverarbeitung erfasst die Exception Rate (Ausnahmequote) den relativen Anteil aller Prozessläufe, bei denen der Standardpfad (Happy Path) aufgrund einer Anomalie verlassen werden musste und ein strukturiertes Exception Handling ausgelöst wurde [13, vgl. Abschn. 3.3] [1]. Für eine präzise Systemanalyse ist in der Softwaretheorie eine differenzierte Betrachtung nach den zugrunde liegenden Fehlertypen zwingend erforderlich:

1. **Business Exception Rate:** Misst fachliche Prozessabweichungen (z. B. rechnerische Diskrepanzen oder unvollständige Belegdaten), die eine inhaltliche Prüfung erzwingen. Sie isoliert die inhärente Fehlerhaftigkeit des Datenmaterials vom technischen Vermögen der Software [8] [1, S. 11–12].

2. **System Exception Rate:** Erfasst den Anteil technischer Infrastruktur- und Applikationsfehler. Sie ist ein direkter Gradmesser für die Instabilität der Systemlandschaft oder ein unzureichendes technisches UI-Skripting [8] [1].

Bot Success Rate Die Bot Success Rate (Roboter-Erfolgsquote) dient der isolierten Bewertung der funktionalen Resilienz und Stabilität des RPA-Workflows [13, vgl. Abschn. 3.2]. Im Gegensatz zur STP-Rate, welche durch fachliche Geschäftsfehler oder ungenaue KI-Vorhersagen gemindert wird, misst die Bot Success Rate ausschließlich das technische Durchhaltevermögen des Software-Agenten [8].

Ein Prozesslauf gilt im Sinne dieser Metrik bereits dann als erfolgreich, wenn der Bot transiente Systemfehler über integrierte Retry-Mechanismen autonom abfängt, kontrollierte Rollbacks durchführt oder fehlerhafte Dokumente sauber isoliert in eine Fachbereichs-Warteschlange routet, ohne abzustürzen [9, 8]. Eine niedrige Bot Success Rate deutet auf gravierende Mängel in der softwaretechnischen Implementierung der Try-Catch-Logiken hin und führt zu unkontrollierten Systemabbrüchen, die teure IT-Supporteinsätze nach sich ziehen [1, vgl. S. 11].

Durchlaufzeit Aus Sicht des Geschäftsprozessmanagements ist die Durchlaufzeit (t_{DLZ}) eine zeitbasierte Leistungskennzahl, welche die absolute Zeitspanne misst, die ein Beleg vom physischen oder digitalen Eintritt in das System bis zur finalen Verbuchung im ERP-System benötigt. Bei einer manuellen Rechnungsverarbeitung wird diese Zeit primär durch Liege-, Transport- und manuelle Bearbeitungszeiten dominiert [14, S. 202].

Durch den Einsatz von RPA wird die reine Systemverarbeitungszeit auf wenige Sekunden oder Minuten reduziert [7, S. 298]. Dennoch verbleibt innerhalb hybrider Architekturen

eine zeitliche Varianz: Sobald eine Business Exception eine Human-in-the-Loop-Validierung auslöst, gerät der Bot in einen Wartezustand, bis der menschliche Supervisor aktiv wird [7, S. 296]. Die Durchlaufzeit ist daher ein kritischer Indikator für die prozessuale Effizienz der angebotenen Validierungsschnittstellen der jeweiligen Plattformen[14, vgl. S. 203].

On-Time Payment Rate (Pünktlichkeitsquote) On-Time Payment Rate (Pünktlichkeitsquote) Die On-Time Payment Rate fungiert als strategische Schnittstelle zwischen operationalen IT-Metriken und der übergeordneten Corporate Reputation [15, S. 2–4]. Sie misst den prozentualen Anteil der Lieferantenrechnungen, die unter Ausnutzung von Skontoeffekten innerhalb der vertraglich vereinbarten Zahlungsfrist erfolgreich angewiesen werden [12, vgl. S. 14].

In der betriebswirtschaftlichen Praxis besteht eine direkte Kausalität zwischen IT-gestützter Prozessbeschleunigung und dem Imagegewinn eines Unternehmens gegenüber externen Marktpartnern [16, 15]. Da eine Reduktion der Durchlaufzeiten das Risiko von Mahngebühren minimiert und gleichzeitig die Liquiditätsplanung präzisiert, dient die theoretische On-Time Payment Rate als valider Proxy, um den qualitativen Reputationsgewinn einer stabilen RPA-Infrastruktur messbar zu machen [17].

2.6. Rechnungsverarbeitung

Die Rechnungsverarbeitung bildet als Teilbereich der Kreditorenbuchhaltung den kontrollintensiven Abschluss des P2P-Zyklus [14, 2]. Als hochgradig strukturierter, repetitiver Supportprozess mit hohem Transaktionsvolumen stellt sie ein primäres Anwendungsfeld für die RPA dar [1, S. 7]. Das prozessuale Referenzmodell gliedert sich in sequentielle Phasen vom Multi-Channel-Rechnungseingang über die Erfassung, die formale und inhaltliche Prüfung (Drei-Wege-Abgleich), optionale Klärungsworkflows und die Freigabe bis hin zur Buchung im ERP-System und der Langzeitarchivierung [14, 2].

Für eine smarte End-to-End-Automatisierung via RPA erfordert die unstrukturierte Natur von Rechnungsbelegen den Einsatz von KI-gestütztem IDP [4, S. 2]. Während klassische Optical Character Recognition (OCR)-Verfahren an variierenden Layouts scheitern, ermöglichen CNNs und NLPs eine kontextbasierte Datenextraktion [4, S. 2-3]. Das informationstechnologische Optimum bildet das Straight-Through Processing über einen automatisierten Drei-Wege-Abgleich zwischen Rechnung, Bestellung und Wareneingang, welcher die administrativen Prozesskosten erheblich senkt [4, S. 3] [3].

In der Automatisierungspraxis bestimmen jedoch Prozesskomplexität und Sonderfälle maßgeblich den Implementierungsaufwand [1, S. 11-12]. Weichen Daten außerhalb systemseitiger Toleranzgrenzen ab, schlägt die Dunkelbuchung fehl, was die Implementierung robuster Exception-Handling-Mechanismen auf den eingesetzten RPA-Plattformen erfordert. Dabei muss das System softwareseitig zwischen technischen Systemfehlern (System Exceptions, z. B. API-Timeouts oder ERP-Systemausfälle) und fachlichen Geschäftsfehlern (Business Exceptions, z. B. Preisabweichungen oder unvollständige Belegdaten nach § 14 UStG [10, vgl. Art. 23]) differenzieren [8]. Der Aufwand für die Modellierung dieser Fehlerpfade und die Bereitstellung von Schnittstellen zur menschlichen Validierung HITL beeinflusst die Integrationsfähigkeit sowie die letztliche Wirtschaftlichkeit der RPA-Lösungen, gemessen an Metriken wie dem ROI, der Amortisationsdauer und den erzielten FTE-Einsparungen [1, S. 11-12] [11].

3. Ausgewählte Technologien

Das folgende Kapitel stellt die ausgewählten Technologien vor, die für die Umsetzung und Evaluierung der automatisierten Kreditorenprozesse herangezogen werden. Die Auswahl umfasst das zentrale ERP-Zielsystem sowie verschiedene Automatisierungsplattformen, die sich über unterschiedliche Software-Paradigmen von Low-Code bis Pro-Code erstrecken. Die Analyse fokussiert sich jeweils auf deren Systemarchitektur, die integrierten Funktionen zur Dokumentenextraktion sowie die plattformeigenen Mechanismen zur Fehlerbehandlung.

3.1. Dolibarr ERP (Document Understanding)

Abbildung 1: Dolibarr ERP System. [18]

Dolibarr repräsentiert ein quelloffenes, modular aufgebautes Enterprise Resource Planning-System [19, S. 1–2], das auf einer klassischen LAMP- beziehungsweise WAMP-Architektur basiert [18]. Als transaktionale Datenbasis und administratives Zielsystem für automatisierte Kreditorenprozesse ist das System strikt prozessorientiert aufgebaut. Es verwaltet sowohl Stammdaten als auch transaktionale Belege in einer relationalen MySQL- oder PostgreSQL-Datenbank [18]. Die informationstechnische Herausforderung bei der Einbindung von Document-Understanding-Komponenten in diese Systemumgebung betrifft die nahtlose Verknüpfung unstrukturierter Belegdaten mit den relationalen Tabellenstrukturen der Finanz- und Lagerhaltungsmodule. Dies betrifft in der Praxis insbesondere die systemeigenen Tabellen für Lieferantenrechnungen und Bestellungspositionen [19, vgl. Abb. 1].

In automatisierten IT-Architekturen fungiert ein solches ERP-System als zentraler Validierungspunkt, an den extrahierte Daten übergeben werden müssen. Da die Plattform über eine standardisierte REST-Schnittstelle verfügt [18], erfordert die Interaktion die clientseitige Generierung und Übertragung strukturierter JSON-Objekte [4, S. 4] [7, S. 297]. Die Integration von Automatisierungssoftware wird dabei maßgeblich durch die serverseitige Validierungslogik des ERP-Systems beeinflusst. Jedes informationstechnische Fehlschlagen einer API-Verbindung sowie jede inhaltliche Ablehnung eines Datensatzes, beispielsweise beim Auftreten von Duplikaten durch identische Rechnungsnummern oder bei Verletzungen der relationalen Integrität, führt zu einer serverseitigen Fehlermeldung und triggert im aufrufenden System eine Exception [8, 9]. Diese deterministischen Validierungsmechanismen machen relationale Open-Source-ERP-Systeme zu einem etablierten Gegenstand in der Wirtschaftsinformatikforschung, um die Integrationsfähigkeit, Schnittstellenkompatibilität und Fehlertoleranz von komplementären Automatisierungslösungen theoretisch und praktisch zu analysieren [19].

3.2. UiPath

Abbildung 2: UiPath RPA. [20]

UiPath stellt eine Enterprise-RPA-Plattform dar, die in der industriellen Praxis zu den Marktführern gehört und im Gartner Magic Quadrant for Robotic Process Automation kontinuierlich als Leader geführt wird [21]. Das technologische Fundament der Plattform wird vom Microsoft .NET-Framework gebildet, das eine standardisierte Laufzeitumgebung für verwalteten Code bereitstellt [22]. Die informationstechnische Architektur der Plattform unterteilt sich in die Komponenten Studio als integrierte Entwicklungsumgebung, Orchestrator als zentrale cloudbasierte oder On-Premise-Steuerungseinheit und Robots als ausführende Software-Einheiten [21]. UiPath steht paradigmatisch für einen klassischen, grafisch orientierten Low-Code-Entwicklungsansatz, bei dem die Automatisierungsprozesse im Studio überwiegend visuell per Drag-and-Drop über vorgefertigte Aktivitätsbausteine modelliert werden [23].

UiPath verfügt über standardisiertes, visuelles Fehlerbehandlungs-Framework, das im Robotic Enterprise Framework (RE-Framework) materialisiert ist [23]. Dieses baut direkt auf der strukturierten try-catch-finally-Logik und der objektorientierten Hierarchie von .NET auf, bei der das Framework eine strikte Typisierung aller von der zentralen Basisklasse System.Exception abgeleiteten Ausnahmen erzwingt [22]. Die Fehlerbehandlung wird über grafische Try-Catch-Blöcke realisiert, die eine strikte Trennung zwischen System Exceptions (z. B. ein unerwarteter Systemabsturz des ERP-Clients) und Business Exceptions (z. B. eine mathematische Abweichung beim Drei-Wege-Abgleich) direkt im Plattformmodell erzwingen [8]. Ergänzend bringt die Suite eigene Konstrukte für technische Fehlerpfade mit, wie die isolierte Retry-Scope-Aktivität oder die Konfiguration eines zentralen, automatisierten Wiederholungsversuchs (Retry) über die Warteschlangen-Steuerung des Orchestrators [23].

Für die KI-gestützte Dokumentenextraktion greift die Plattform auf das integrierte Werkzeug UiPath Document Understanding zurück. Dieses System kombiniert klassische OCR-Engines mit maschinellen Lernmodellen und spezialisierten Large Language Models (LLMs), die direkt über das AI Center innerhalb des Orchestrators verwaltet und trainiert werden. Die extrahierten Rechnungsdaten werden zusammen mit einem mathematischen Confidence Score je Datenfeld ausgegeben. Unterschreitet dieser Konfidenzwert eine im Bot definierte Schwelle, greift der integrierte Action Center-Mechanismus, welcher die fehlerhafte Rechnung in eine isolierte Human-in-the-Loop-Warteschlange umleitet. Dort kann ein menschlicher Auditor die Daten über den plattformeigenen Validation Station-Bildschirm manuell korrigieren und verifizieren. [24]

3.3. Microsoft Power Automate

Abbildung 3: Microsoft Power Automate. [25]

Microsoft Power Automate ist eine cloudbasierte Low-Code-Automatisierungsplattform innerhalb der Microsoft-Power-Platform-Umgebung. Flows werden in der Regel als Cloud Flows modelliert und nutzen standardisierte Konnektoren, um mit Cloud-Diensten wie Outlook, SharePoint oder Dataverse zu interagieren [26, 27]. Für die Anbindung lokaler oder hybrider Szenarien kann Power Automate zudem auf Gateway-gestützte Verbindungen zurückgreifen, um Desktop- oder On-Premises-Systeme in automatisierte Abläufe einzubinden [26].

Zur erweiterten Administration und lokalen Systemsteuerung kann PowerShell eingesetzt werden. Microsoft beschreibt PowerShell als eine Skriptumgebung, die zwischen nicht-abbrechenden und abbrechenden Fehlern unterscheidet; nicht-abbrechende Fehler stoppen die Ausführung nicht automatisch, während abbrechende Fehler die Pipeline beenden können. Dieses Verhalten kann über Variablen wie $ErrorActionPreference oder durch ErrorAction Stop beeinflusst werden, wodurch Fehler deklarativ gesteuert werden können. [28]

Die Fehlerbehandlung in Power Automate erfolgt ebenfalls deklarativ. Microsoft empfiehlt die Konfiguration von Run after-Einstellungen sowie den Einsatz von Scopes und der Terminate-Aktion, um nachgelagerte Schritte abhängig vom Ergebnis vorheriger Aktionen auszuführen. Ein Folgeschritt kann so konfiguriert werden, dass er nur bei bestimmten Zuständen wie fehlgeschlagen, übersprungen oder zeitüberschritten ausgeführt wird. [26]

Für die Dokumentenerkennung und -extraktion kann der AI Builder mit dem vorgefertigten Modell für die Rechnungsverarbeitung verwendet werden. Microsoft beschreibt dieses Modell als ein Werkzeug zur Extraktion wichtiger Rechnungsdaten; die extrahierten Felder werden dabei mit Konfidenzwerten versehen [29]. Auf dieser Basis lassen sich technische Fehler oder unsichere Extraktionsergebnisse in nachgelagerte Genehmigungs- oder Prüfprozesse überführen [29, 27]. Adaptive Cards in Microsoft Teams ermöglichen dabei eine strukturierte Interaktion mit menschlichen Sachbearbeitern, sodass ein Human-in-the-Loop-Ansatz umgesetzt werden kann [27].

3.4. Sema4.ai (Nanonets)

Abbildung 4: Sema4.ai. [30]

Sema4.ai verfolgt ein pro-code-orientiertes Paradigma und basiert auf dem Open-Source-Framework Robocorp; die Automatisierungslogik wird dabei in Python implementiert [31]. Python ist eine interpretierte Programmiersprache mit klarer Syntax, dynamischer Typisierung und einem umfangreichen Ökosystem an Bibliotheken [32]. Im Unterschied zu visuellen Low-Code-Oberflächen wird die Prozesslogik textbasiert als Quellcode formuliert; typische Komponenten beinhalten Bibliotheken für Workflow-Strukturierung und Browserautomatisierung, die in Robocorp-basierten Projekten verwendet werden können [31, 33].

Die Fehlerbehandlung erfolgt programmatisch über das native try-except-Modell von Python, wobei Ausnahmen als Objekte behandelt werden und dadurch benutzerdefinierte Fehlerklassen zur feingranularen Behandlung möglich sind. Entwickler können Retry-Strategien, eigene Statuslogiken und parametrisierte Fehlerbehandlungsroutinen direkt im Code implementieren, wodurch technische Infrastrukturfehler und fachliche Regelverletzungen bereits auf Programmebene getrennt werden können. [32, 31]

Für die kognitive Belegverarbeitung wird in Robocorp-Setups häufig ein externer Dokumenten-KI-Dienst angebunden; ein Beispiel hierfür ist die Integration über REST-APIs von Spezialanbietern, die extrahierte Metadaten als strukturiertes JSON zurückgeben. Solche Dienste liefern in der Regel Konfidenzwerte für extrahierte Felder, und bei Unterschreiten definierter Schwellwerte können die Belege einem manuellen Prüfprozess zugeführt werden, bevor validierte Daten vom Automatisierungs-Workflow übernommen werden. Lokale Nutzerinteraktionen lassen sich außerdem durch begleitende Komponenten oder Assistenz-UI-Module in den Workflow integrieren.[31, 34]

Dieser code-zentrierte Ansatz erfordert in der Regel höheres technisches Entwicklungs-Know-how und initiale Implementierungsaufwände, bietet jedoch maximale Flexibilität bei der Modellierung komplexer Fallback-Mechanismen und erlaubt eine sehr feingranulare Steuerung von Fehlerfällen, was langfristig die Wartbarkeit bei anspruchsvollen Automatisierungsszenarien verbessern kann [31].

4. Verwandte Arbeiten

Dieses Kapitel ordnet die vorliegende Arbeit in den Stand der Forschung ein. Über eine Literaturrecherche wurden Arbeiten ausgewählt, die im Kern dasselbe Ziel verfolgen: die Automatisierung der Rechnungsverarbeitung mit Robotic Process Automation, oft ergänzt um KI-gestützte Extraktion, oder den Vergleich von RPA-Plattformen. Die ausgewählten zehn Arbeiten werden im Folgenden nach inhaltlichen Schwerpunkten gruppiert besprochen. Anschließend werden Gemeinsamkeiten und Unterschiede zur eigenen Arbeit herausgearbeitet und in Tabelle 1 verdichtet.

4.1. Automatisierung der Rechnungsverarbeitung mit RPA

Die größte Gruppe verwandter Arbeiten setzt RPA ein, um den Rechnungseingang von der manuellen Bearbeitung zu lösen. Santos und Trigo schlagen eine RPA-Lösung vor, die Rechnungen ausliest und in ein ERP-System überträgt, und stehen damit der vorliegenden Arbeit thematisch am nächsten [35]. Desai et al. setzen Automation Anywhere zusammen mit Computer Vision und maschinellem Lernen ein und begründen den Automatisierungsbedarf über eine Marktbefragung [36]. Rohaime et al. kombinieren RPA mit KI und OCR und berichten für ihren Datensatz eine Genauigkeit von 100 % bei einer Verarbeitungszeit unter 30 Sekunden je Rechnung [37]. Bei allen drei Arbeiten ist gemeint, dass sie eine einzelne Plattform oder eine selbst gebaute Lösung umsetzen und vorrangig Genauigkeit und Zeitersparnis berichten. Eine systematische Behandlung von Ausnahmefällen oder ein Plattformvergleich findet nicht statt.

4.2. Intelligente Dokumentenverarbeitung und KI-Erweiterung

Tater et al. beschreiben ein produktiv eingesetztes End-to-End-System zur Rechnungsverarbeitung, das KI-gestützte Extraktion mit regelbasierten Validierungen kombiniert. Das Exception Handling wird über dynamische Konfidenzschwellen gesteuert: Werden diese unterschritten, erfolgt eine Aussteuerung an eine Human-in-the-Loop-Schnittstelle zur manuellen Korrektur. Mit einer Erfolgsquote von 76 % bei 80.000 Rechnungen liefert die Arbeit einen empirischen Praxisbeweis für das Potenzial der Automatisierung. [38]

Obwohl Tater et al. identische Konzepte nutzen (wie HITL und Confidence Scores), betrachten sie lediglich eine proprietäre, individuell programmierte Einzellösung. Ein systematischer Plattformvergleich kommerzieller RPA-Lösungen (wie UiPath, Power Automate oder Sema4.ai) sowie die Quantifizierung des Implementierungsaufwands für Fehlerpfade finden nicht statt. [38]

4.3. Vergleich von RPA-Plattformen

Eine dritte Gruppe vergleicht RPA-Werkzeuge unmittelbar. Sindhuja et al. stellen UiPath, Automation Anywhere und Robocorp gegenüber und betrachten Funktionsumfang, Skalierbarkeit, Bedienbarkeit und Leistung [39]. Da Robocorp die technische Basis von Sema4.ai bildet, liegt diese Arbeit dem Plattformvergleich der vorliegenden Arbeit am nächsten. Khan vergleicht UiPath, Automation Anywhere und BluePrism auf ähnliche Weise [40]. Beide Arbeiten stützen sich vorrangig auf eine theoretische Gegenüberstellung von Merkmalen und Preismodellen.

Einen praxisorientierten Ansatz wählen hingegen Bornegrim und Holmquist, die für ihren Vergleich von UiPath, Automation Anywhere und Blue Prism tatsächlich einen realen Prozess in allen drei Tools programmieren. Allerdings handelt es sich bei deren Anwendungsszenario um einen stark vereinfachten Prozess, der lediglich E-Mails und einfache Excel-Tabellen verarbeitet. [41, S. 16]

Ein konkreter buchhalterischer Rechnungsprozess, eine tiefergehende Ausnahmebehandlung (Exception Handling) von System- und Geschäftsfehlern oder gar eine ERP-Buchung

werden in keiner dieser drei Studien untersucht. Genau diese Forschungslücke wird durch das Anwendungsszenario der vorliegenden Arbeit geschlossen.

4.4. Einordnung der eigenen Arbeit

Tabelle 1 stellt die zehn Arbeiten der vorliegenden gegenüber. Bewertet werden der Einsatz von KI- bzw. OCR-Extraktion mit Konfidenzwerten, eine Human-in-the-Loop-Validierung, eine dedizierte Ausnahmebehandlung, die tatsächliche Buchung in ein ERP-System, ein Vergleich mehrerer Plattformen sowie die empirische Absicherung.

Tabelle 1: Einordnung der verwandten Arbeiten (+ vorhanden, o ansatzweise, – nicht behandelt).

Arbeit	**Plattform(en)**	**KI/OCR**	**HITL**	**Excp.**	**ERP**	**Vergl.**	**Test**
Santos & Trigo [35]	ein RPA-Tool	+	–	–	+	–	PoC
Desai et al. [36]	Autom. Anywhere	+	–	–	–	–	Prototyp
Rohaime et al. [37]	ein RPA-Tool	+	–	–	o	–	klein
Tater et al. [38]	ein (proprietär)	+	+	o	+	–	ca. 80.000
Sindhuja et al. [39]	UiPath/AA/Robocorp	–	–	–	–	+	Review
Khan [40]	UiPath/AA/BluePrism	–	–	–	–	+	Review
Bornegrim & Hol. [41]	UiPath/AA/BluePrism	–	–	–	–	+	Prototyp
Diese Arbeit	**UiPath/Power Automate/Sema4.ai**	+	+	+	+	+	Korpus (30)

Die Gegenüberstellung zeigt zwei Forschungsstränge, die bislang getrennt bleiben. Der erste Strang automatisiert die Rechnungsverarbeitung mit RPA und KI, beschränkt sich aber auf eine einzelne Plattform und behandelt die Ausnahmebehandlung allenfalls am Rande. Der zweite Strang vergleicht RPA-Plattformen, tut dies jedoch primär theoretisch anhand von Merkmalslisten oder – wie im Fall von Bornegrim und Holmquist [41] – anhand stark vereinfachter Prototypen, die weder intelligente Dokumentenverarbeitung noch eine ERP-Integration umfassen. Die vorliegende Arbeit verbindet beide Stränge: Sie setzt denselben Rechnungsprozess auf drei Plattformen um, darunter mit Sema4.ai eine auf Robocorp basierende Lösung, vergleicht sie an einem identischen Rechnungskorpus und legt den Schwerpunkt ausdrücklich auf die Behandlung von System- und Geschäftsfehlern, die Human-in-the-Loop-Validierung und die tatsächliche Buchung in ein ERP-System.

Eine Arbeit, die exakt dasselbe leistet, wurde nicht gefunden. Am nächsten kommen Santos und Trigo, die denselben Prozess umsetzen, jedoch nur auf einer Plattform und ohne Schwerpunkt auf der Ausnahmebehandlung [35], sowie Sindhuja et al., die zwar drei Plattformen einschließlich Robocorp vergleichen, dies aber ohne konkreten Rechnungsprozess und ohne Fehler- oder Validierungsbetrachtung tun [39]. Genau in der Schnittmenge dieser Arbeiten liegt der Beitrag der vorliegenden Arbeit.

Zugleich bleibt die eigene Arbeit in einigen Punkten bei der Bewertung der Ergebnisse zu berücksichtigen. Der Rechnungskorpus ist mit dreißig Rechnungen klein und besteht aus selbsterstellten Rechnungen, während etwa Tater et al. rund 80.000 reale Rechnungen verarbeiten [38]. Die Arbeit bezieht sich nicht aus der Datenmenge, sondern aus dem direkten Vergleich dreier Plattformen an einem identischen Prozess mit ausdrücklichem Fokus auf Ausnahmebehandlung, menschlicher Validierung und realer ERP-Buchung.

5. Methodik

In diesem Kapitel wird das Forschungsdesign beschrieben. Um die Ausnahmebehandlung der ausgewählten RPA-Systeme zu vergleichen, wird ein identischer, KI-gestützter Rechnungsprüfungsprozess auf drei verschiedenen Plattformen implementiert. Zunächst wird das Anwendungsszenario vorgestellt (Abschnitt 5.1). Darauf folgt die Begründung der Plattformauswahl (Abschnitt 5.2). Abschnitt 5.3 beschreibt den konkreten Prozessaufbau, also die einzelnen Schritte und die Fehlerbehandlung. Abschnitt 5.4 legt schließlich die Metriken fest, an denen die drei Implementierungen gemessen werden.

5.1. Vorstellung des Anwendungsszenarios

Als Anwendungsszenario dient die automatisierte Verarbeitung eingehender Lieferantenrechnungen. Dieser Prozess fällt in nahezu jedem Unternehmen an, weist ein hohes Automatisierungspotenzial auf und ist gleichzeitig fehleranfällig [1]. Die Rechnungen gehen als PDF-Dateien in unterschiedlichen Layouts ein; sie müssen automatisiert ausgelesen, validiert und schließlich in ein ERP-System übertragen werden [4, S. 1–2].

Genau an diesen Schnittstellen entstehen häufig Fehler, die sich in zwei Kategorien einteilen lassen: technische Systemfehler (wie ein temporär nicht erreichbarer Dienst) und fachliche Geschäftsfehler (wie ein fehlendes oder unplausibles Datenfeld) [8]. Da im gewählten Szenario beide Fehlerklassen auftreten und jeweils unterschiedliche Behandlungsstrategien erfordern, bietet die Rechnungsverarbeitung eine geeignete Grundlage, um das Exception Handling der betrachteten RPA-Systeme systematisch zu untersuchen [1, S. 11].

5.1.1. Fachlicher Rahmen und eingesetzte Systeme

Den Ausgangspunkt bilden PDF-Rechnungen, die in einem Eingangsverzeichnis bereitgestellt werden. Das inhaltliche Auslesen der Rechnungsdaten erfolgt jeweils über eine KI-gestützte Komponente der betrachteten Plattform. Bei UiPath wird hierfür das hauseigene Document Understanding eingesetzt [24], während bei Microsoft Power Automate die Rechnungsverarbeitung über das vorgefertigte KI-Modell des AI Builders genutzt wird [42]. Da Sema4.ai primär eine Python-basierte Automatisierungsplattform darstellt [43] und keine vergleichbare native Rechnungsanalyse im gleichen Umfang bereitstellt, wird dort ein externes Extraktionsmodell angebunden. Für diese Implementierung wird Nanonets verwendet. Als Ziel- und Buchungssystem wird das quelloffene ERP-System Dolibarr verwendet [19, S. 1–2] [18]. Dort wird für jede Rechnung ein Lieferantenrechnungsbeleg angelegt. Zusätzlich werden die extrahierten Daten in eine Excel-Datei geschrieben, die als Kontroll- und Auswertungsprotokoll dient.

Die Entscheidung für ein frei verfügbares ERP-System dient der Vergleichbarkeit der drei Implementierungen. Ein proprietäres System eines einzelnen Anbieters könnte den Plattformvergleich verzerren, da spezifische Schnittstellen oder Herstellerabhängigkeiten einzelne Lösungen bevorzugen würden. Dolibarr bildet hingegen für alle drei Implementierungen dasselbe Zielsystem, sodass die RPA-Lösungen unter vergleichbaren Bedingungen umgesetzt und bewertet werden können [19, vgl. S. 2].

5.1.2. Zu extrahierende Datenfelder

Aus jeder Rechnung gewinnt der Prozess zwei Arten von Information. Kopfdaten kommen genau einmal je Beleg vor. Positionsdaten liegen als Tabelle vor, deren Länge von Rechnung zu Rechnung schwankt. Tabelle 2 listet die Felder auf.

Tabelle 2: Im Anwendungsszenario ausgelesene Rechnungsfelder.

Ebene	Felder
Kopfdaten (1× je Beleg)	Lieferantenname, Lieferantennummer, Rechnungsnummer, Rechnungsdatum, Fälligkeitsdatum, Gesamtbruttobetrag
Positionsdaten (n× je Beleg)	Anzahl, Artikelnummer, Einzelnettobetrag, Positionsbeschreibung

5.1.3. Statusbasierte Ablagesteuerung

Der Prozess steuert seinen Zustand über ein Ordnersystem, bei dem jede Rechnung während der Verarbeitung mehrere Verzeichnisse durchläuft, deren Name den aktuellen Bearbeitungsstand angibt. Dies macht den Zustand jeder Rechnung jederzeit ablesbar, verhindert eine erneute Verarbeitung bereits bearbeiteter Belege und das Ergebnis der Ausnahmebehandlung wird sichtbar, da jede Fehlerklasse in einem eigenen Ordner landet. Tabelle 3 beschreibt die Zustände.

Tabelle 3: Statusverzeichnisse zur Ablaufsteuerung und Ergebnisablage.

Verzeichnis	Bedeutung
001 – Unbearbeitet	Eingang neuer, noch nicht verarbeiteter PDF-Rechnungen.
002 – In Bearbeitung	Rechnung wird gerade verarbeitet; dient als Sperre gegen Doppelverarbeitung.
003 – Verarbeitet	Erfolgreich gebucht; ausgelesener und im ERP berechneter Bruttobetrag stimmen überein.
007 – Manuell überprüfen	Im Human-in-the-Loop abgebrochen; die Begründung hängt am Dateinamen.
008 – Bruttobetrag ungleich	Gebucht, aber der erwartete und der berechnete Bruttobetrag weichen ab.
009 – Fehler	Technischer System- bzw. Laufzeitfehler während der Verarbeitung.

5.2. Auswahl der RPA-Plattformen

Robotic Process Automation ist ein Sammelbegriff für Werkzeuge, die andere Anwendungen über deren Benutzeroberfläche bedienen, so wie es ein Mensch täte [44]. Der Markt bietet dafür eine Bandbreite an Werkzeugen, von grafischen Low-Code-Suiten bis zu code-orientierten und zunehmend agentenbasierten Frameworks. Für den Vergleich wurden drei Plattformen ausgewählt, die unterschiedliche Ansätze vertreten, beim Anwendungsfall aber vergleichbar bleiben.

5.2.1. Auswahlkriterien

Vier Kriterien lagen der Auswahl zugrunde:

- **Marktrelevanz:** Die Plattform spielt im professionellen Umfeld eine Rolle, sodass die Ergebnisse praktischen Bezug haben.
- **Unterschiedliches Paradigma:** Low-Code, Cloud-integriert und Code-First sollen abgedeckt sein, denn gerade darin unterscheiden sich die Mechanismen der Fehlerbehandlung (siehe Abschnitt 3).

- **KI-gestützte Extraktion:** Die Plattform stellt ein Intelligent Document Processing bereit oder kann es über eine Schnittstelle anbinden, jeweils mit einem Konfidenzwert je Feld.
- **Zugänglichkeit:** Eine Community-, Test- oder Entwicklerlizenz erlaubt die vollständige Umsetzung des Szenarios.

5.2.2. Gegenüberstellung

Tabelle 4 stellt die drei ausgewählten Plattformen (siehe Abschnitt 3) entlang der für das Exception Handling wichtigen Eigenschaften gegenüber.

Tabelle 4: Gegenüberstellung der ausgewählten RPA-Plattformen.

	UiPath	**Power Automate**	**Sema4.ai / Robocorp**
Paradigma	Low-Code, grafisch (Studio)	Low-Code, cloud-integriert	Code-first / agentisch (Python)
KI-Extraktion	Document Understanding (eingebaut)	AI Builder – Rechnungsverarbeitung (eingebaut)	Nanonets (externe REST-API)
Validierung (HITL)	Validation Station (eingebaut)	Bedingungen + manuelle Genehmigung	Eigenbau (Dialog + Nanonets-Review)
Fehlerbehandlung	Try-Catch, Orchestrator-Retry	„Ausführen-nach“, Bedingungen	try/except, eigene Retry-Logik
Ausführung	Desktop/Robot + Orchestrator	Cloud (Power Platform)	Lokal/Worker + Control Room

Die Auswahl stellt drei verschiedene Philosophien der Automatisierung nebeneinander. Der spätere Vergleich zeigt damit nicht nur graduelle Unterschiede ähnlicher Werkzeuge, sondern unterschiedliche Wege, Fehler und Ausnahmen zu behandeln.

5.3. Prozessaufbau

Der fachliche Ablauf ist auf allen drei Plattformen gleich. Er unterscheidet sich nur in der technischen Umsetzung der einzelnen Schritte und besonders in der Fehlerbehandlung. Dieser Abschnitt beschreibt den Aufbau plattformübergreifend.

Überblick über den Ablauf Abbildung 5 zeigt das Szenario im Überblick. Eine PDF-Rechnung wird ausgelesen. Ist die Extraktion unsicher, folgt eine manuelle Prüfung. Anschließend wird im ERP gebucht und je nach Ergebnis in das passende Statusverzeichnis abgelegt. Den vollständigen Ablauf mit allen Fehlerverzweigungen beschreiben die folgenden Abschnitte.

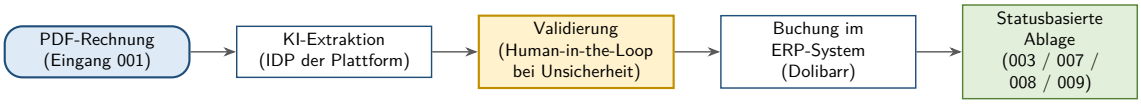


Abbildung 5: Überblick über das Anwendungsszenario der KI-gestützten Rechnungsverarbeitung.

Start und Anmeldung am ERP-System Zu Beginn des Prozesses wird das Quellverzeichnis auf unbearbeitete Eingangsbelege (PDF-Rechnungen) überprüft. Liegen keine Dokumente vor, wird der Prozesslauf umgehend regulär beendet. Andernfalls erfolgt die Authentifizierung am Zielsystem (ERP-/Buchungssystem).

Bereits an dieser Stelle greift der erste Mechanismus der technischen Fehlerbehandlung: Vor dem eigentlichen Login-Versuch prüft der Prozess in einer definierten Schleife (Mehrfachversuche mit Warteintervallen), ob der Server des Zielsystems erreichbar ist. Bleibt der Server nach allen Versuchen antwortlos, bricht der Prozesslauf kontrolliert mit einer entsprechenden Fehlermeldung ab. Diese strikte Trennung zwischen einer temporären Verbindungsstörung und einem dauerhaften Systemausfall ist ein durchgängiges Prinzip der Prozessarchitektur.

Schleife über die Rechnungen und KI-Extraktion Nach erfolgreicher Anmeldung am Zielsystem werden die gefundenen Belege sequenziell abgearbeitet. Für jedes Dokument wird der Status im Dateisystem initial von „unbearbeitet“ auf „in Bearbeitung“ umgesetzt. Dieser Statuswechsel dient als Sperrmechanismus, um eine redundante Doppelbearbeitung zu verhindern und den aktuellen Verarbeitungsstatus transparent zu machen.

Anschließend wird das Dokument an eine KI-gestützte Extraktionskomponente via API übermittelt. Auch dieser Schritt ist durch eine Retry-Schleife abgesichert, die zwei Fehlerklassen differenziert:

1. **Temporäre Verbindungsprobleme (z. B. Timeouts, Serverfehler 5xx):** Der Prozess initiiert einen erneuten Aufrufversuch.
2. **Permanente Konfigurations-/API-Fehler (z. B. ungültige Authentifizierung, falsche Modell-IDs, Client-Fehler 4xx):** Der Prozess bricht die Verarbeitung sofort ab, da erneute Versuche systemisch wirkungslos wären.

Scheitert die Datenextraktion final, wird der Beleg in ein systemisches Fehlerverzeichnis verschoben. Im Erfolgsfall liefert die KI-Komponente für jedes extrahierte Datenfeld den erkannten Text sowie einen mathematischen Konfidenzwert. Mehrseitige Dokumente werden konsolidiert, sodass sowohl Kopfdaten als auch die einzelnen Positionstabellen extrahiert werden.

Konfidenzprüfung und Human-in-the-Loop Der weitere Prozessweg wird deterministisch durch die extrahierten Konfidenzwerte bestimmt. Unterschreitet auch nur ein einziges Kopffeld oder eine Tabellenzelle den definierten Schwellenwert für die Mindestkonfidenz, gilt die Extraktion als unsicher. Der Prozess initiiert daraufhin einen Human-in-the-Loop-Schritt: Er stellt den Beleg in einer Validierungsoberfläche bereit und pausiert die automatisierte Verarbeitung.

Für die menschliche Prüfung ergeben sich zwei Pfade:

1. **Korrektur und Freigabe:** Der Prüfer korrigiert die unsicheren Werte manuell und gibt den Datensatz frei. Der Prozess übernimmt die verifizierten Daten, markiert sie als „geprüft“ und setzt die Verarbeitung direkt fort. Eine erneute Konfidenzprüfung entfällt in diesem Fall.
2. **Abbruch der Validierung:** Der Prüfer bricht die Bearbeitung ab. Der Prozess verschiebt den Beleg in ein dafür vorgesehenes Ausnahmeverzeichnis und ergänzt den Dateinamen um die angegebene Abbruchbegründung. Der Vorgang wird damit als fachliche Ausnahme dokumentiert und von der automatischen Weiterverarbeitung ausgeschlossen.

Liegen alle Konfidenzwerte ab Werk über dem Schwellenwert, wird die Validierungsstation automatisch übersprungen. Dieser Schritt dient somit primär der Abhandlung von geschäftlichen Ausnahmen, die systemseitig nicht zweifelsfrei validiert werden konnten.

Protokoll und Buchung im ERP-System Die verifizierten Daten werden zunächst in einem lokalen Kontrollprotokoll (z. B. Tabellenkalkulation oder strukturierte Log-Datei) dokumentiert. Dieses Protokoll führt pro Tabellenposition eine Zeile auf, wobei die Kopfdaten redundant mitgeführt werden.

Sowohl die Protokollierung als auch der anschließende Buchungsvorgang sind vollständig in einen Laufzeit-Fehlerabfangblock (try/except) eingebettet, um unvorhergesehene Systemabstürze abzufangen. Im Zielsystem wird daraufhin ein neuer Kreditorbeleg angelegt. Der Prozess prüft, ob der Stammdatensatz des Lieferanten bereits existiert, legt diesen falls nötig automatisch neu an, trägt die relevanten Kopfdaten (Rechnungsnummer, Beleg- und Fälligkeitsdatum) ein, speichert den Belegentwurf und erfasst im Anschluss jede extrahierte Position einzeln.

Bruttobetrag-Vergleich und Ergebnisablage Nach der Erfassung aller Positionen erfolgt ein mathematischer Abgleich der Beträge. Der Prozess liest den vom Zielsystem automatisch berechneten Gesamtbruttobetrag aus und transformiert ihn in ein einheitliches numerisches Format (Bereinigung von Tausendertrennzeichen, Standardisierung des Dezimaltrennzeichens). Dieser Wert wird direkt mit dem von der KI extrahierten bzw. manuell validierten Gesamtbetrag verglichen.

Aus diesem Abgleich resultieren drei finale Ablagestatus:

- **Beträge gleich:** Die Buchung ist fachlich konsistent. Der Beleg wird in das Verzeichnis für erfolgreich verarbeitete Dokumente verschoben.
- **Beträge ungleich:** Es liegt ein fachlicher Fehler (Business Exception) vor. Der Prozess verschiebt das Dokument in ein spezifisches Abweichungsverzeichnis.
- **Laufzeitfehler:** Tritt während der Verarbeitung ein unerwarteter technischer Fehler auf, fängt die Ausnahmebehandlung diesen ab und verschiebt den Beleg in das technische Fehlerverzeichnis.

Jedes Dokument endet somit deterministisch in genau einem von vier definierten Zuständen: erfolgreich verarbeitet, fachlich abweichend, manuell zurückgestellt/abgebrochen oder technisch gescheitert. Diese Struktur ermöglicht im Nachgang eine direkte statistische Auswertung und Metrikerfassung nach Fehlertypen.

Abschluss Nach der finalen Verarbeitung eines Belegs kehrt der Prozess auf eine definierte Ausgangsseite des Zielsystems zurück. Der weitere Ablauf wird durch den Zustand des Eingangsverzeichnisses gesteuert: Befinden sich dort noch weitere unbearbeitete Belege, navigiert der Prozess zurück zur Erstellungsmaske und startet die nächste Iteration. Sind keine Dokumente mehr vorhanden, führt der Prozess eine ordnungsgemäße Abmeldung (Logout) vom Zielsystem durch und beendet den Gesamtlauf in einem definierten, sauberen Zustand.

Gesamtdarstellung Abbildung 6 fasst den modularisierten Ablauf des automatisierten Verarbeitungsprozesses grafisch zusammen. Die farbliche Kodierung der Knoten spiegelt dabei die jeweilige Funktion im Systemaufbau wider:

- **Blaue Knoten** repräsentieren die regulären, automatisierten Prozess- und Verarbeitungsschritte.
- **Gelbfarbene Rauten** kennzeichnen logische Entscheidungen und Verzweigungen auf Basis von Systemantworten oder Datenprüfungen.
- Der **gelbfarbene Kasten** hebt den Human-in-the-Loop-Eingriff zur manuellen Verifikation bei unzureichender Datenqualität hervor.

- Der **grüne Knoten** visualisiert die erfolgreiche finale Ablage konsistenter Belege.
- **Rote Knoten** kennzeichnen die kontrollierte Ablage in den jeweiligen Ausnahme-, Abweichungs- oder Fehlerverzeichnissen.

Der gestrichelte Rahmen markiert den iterativen Teilbereich (Schleife), den der Prozess für jeden identifizierten Eingangsbeleg sequenziell durchläuft. Jede Iteration endet deterministisch in genau einem der definierten Endzustände, bevor die nächste Datei geladen oder der Gesamtlauf ordnungsgemäß beendet wird.

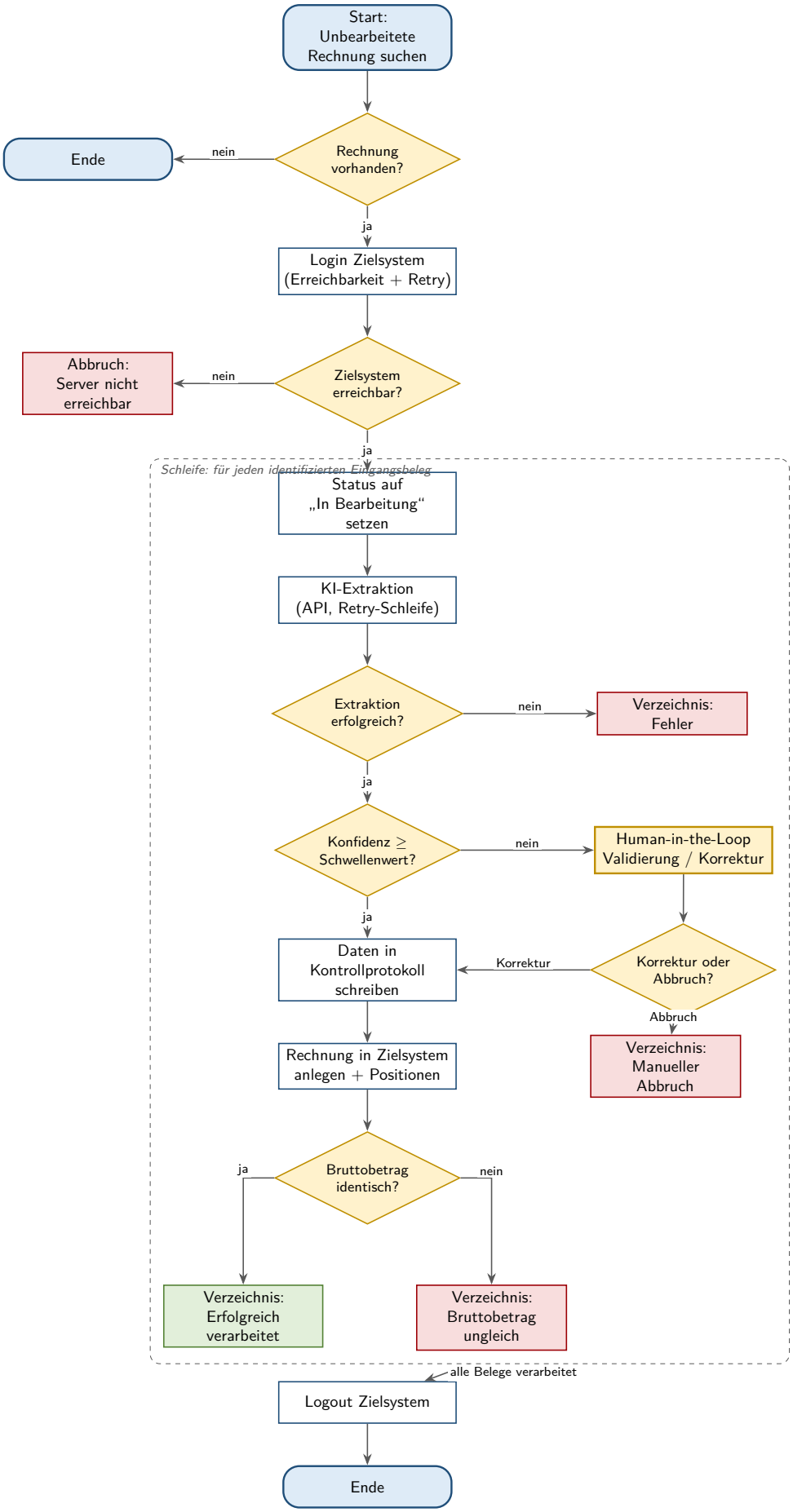


Abbildung 6: Abstrahierter Prozessaufbau der KI-gestützten Rechnungsverarbeitung mit integrierten Verzweigungen zur Fehlerbehandlung und statusbasierter Ergebnisablage.

5.4. Auswahl der Metriken

Zur Bewertung der vorgestellten RPA-Systeme werden mehrere Metrikgruppen herangezogen. Sie ordnen den Implementierungsaufwand, die Prozessleistung und den praktischen Nutzen der betrachteten RPA-Plattformen vergleichbar ein. Die Metriken orientieren sich an den Forschungsfragen der Arbeit und gliedern sich in vier Bereiche: Wirtschaftlichkeit, Prozesseffizienz und Human-in-the-Loop-Validierung, Exception-Handling sowie potenzieller Imagegewinn.

Die Wirtschaftlichkeitsmetriken umfassen unter anderem Return on Investment, Amortisationsdauer, Full-Time Equivalent und Durchlaufzeit. Auf eine tiefgehende Wirtschaftlichkeitsanalyse wird jedoch verzichtet, da hierfür reale Unternehmensdaten, konkrete Kostenstrukturen und belastbare Einsparungswerte fehlen. Stattdessen werden diese Kennzahlen auf einer realistischen Grundlage abgeschätzt und qualitativ eingeordnet, um die wirtschaftliche Relevanz der Automatisierung nachvollziehbar zu machen.

Für die Bewertung der Prozesseffizienz und der Human-in-the-Loop-Validierung dienen die Straight-Through-Processing-Rate, die Exception Rate und die Bot Success Rate. Das Exception-Handling selbst wird anhand ausgewählter Kriterien verglichen, etwa Fehlererkennung, Retry-Konfigurierbarkeit, Workflow-Handling, Ordner-Routing, Validierungsstationen und Logging. Zusätzlich werden qualitative Aspekte eines möglichen Imagegewinns betrachtet, beispielsweise eine verbesserte Prozesswahrnehmung, pünktlichere Zahlungen und eine höhere Transparenz des Rechnungsverarbeitungsprozesses.

5.4.1. Wirtschaftlichkeitsmetriken

Return on Investment

$$\text{ROI} = \frac{\text{Gesamtnutzen} - \text{TCO}}{\text{TCO}} \times 100 \tag{1}$$

Wobei *T*otal Cost of Ownership TCO alle anfallenden Kosten umfasst, darunter Lizenzgebühren, Entwicklungsaufwand und laufende Wartungskosten.

Amortisationsdauer Sie gibt an, nach welchem Zeitraum die Implementierungskosten durch die erzielten monatlichen Einsparungen vollständig gedeckt sind [13, vgl. Abschn. 3]:

$$t_{\text{Amort.}} = \frac{\text{Implementierungskosten}}{\text{Monatliche Einsparungen}} \tag{2}$$

Full-Time Equivalent Durch die FTE-Einsparung wird der durch die Automatisierung freigesetzte Personalaufwand in finanziellen Kennzahlen messbar gemacht [11, vgl. Abschn. B]:

$$\text{FTE-Einsparung} = \text{Eingesparte Stunden} \times \text{Vollkosten-Stundensatz} \tag{3}$$

Die eingesparten Stunden ergeben sich aus der Differenz des manuellen Bearbeitungsaufwands je Rechnung und dem verbleibenden manuellen Aufwand nach Einführung der RPA-Lösung, multipliziert mit dem jährlichen Rechnungsvolumen.

5.4.2. Prozesseffizienzmetriken und HITL-Validierung

Zur Bewertung der Prozesseffizienz und der HITL-Validierung werden drei Kennzahlen eingesetzt, die sich unmittelbar aus der Ordnerstruktur ableiten lassen. Die Ordnerstruktur (001–009) dient dabei als Messinstrument: Über das Verschieben der Rechnungen in die jeweiligen Zielordner lässt sich ermitteln, ob eine Rechnung erfolgreich durchgelaufen ist oder ein Fehler aufgetreten ist.

Straight-Through Processing Rate Die STP-Rate misst den Anteil der Rechnungen, die den gesamten Prozess ohne jeden manuellen Eingriff durchlaufen [12, vgl. S. 3–5]. Im Kontext entspricht das allen Rechnungen, die in Ordner 003 (*Verarbeitet*) abgelegt wurden:

$$\text{STP-Rate} = \frac{\text{Rechnungen in Ordner 003}}{\text{Gesamtrechnungen}} \times 100 \tag{4}$$

Exception Rate Die Exception Rate ist das Gegenstück zur STP-Rate. Sie erfasst den Anteil der Rechnungen, bei denen eine Ausnahmebehandlung ausgelöst wurde [13, vgl. Abschn. 3]:

$$\text{Exception-Rate} = \frac{\text{Rechnungen außerhalb Ordner 003}}{\text{Gesamtrechnungen}} \times 100 \tag{5}$$

Ein Vorteil dieser Metrik im vorliegenden Kontext ist die Möglichkeit zur Aufschlüsselung nach Fehlertypen: Rechnungen im Ordner 007 entsprechen abgebrochenen HITL-Validierungen, Ordner 008 enthält Betragsabweichungen (*Business Exception*) und Ordner 009 technische Fehler (*System Exception*). Diese Differenzierung ermöglicht eine direkte Beantwortung der Teilforschungsfrage 3.

Bot Success Rate Die Bot Success Rate misst den Anteil der Prozessläufe, die ohne technischen Abbruch abgeschlossen wurden [13, vgl. Abschn. 4]:

$$\text{Bot Success Rate} = \frac{\text{Läufe ohne Routing nach Ordner 009}}{\text{Gesamtläufe}} \times 100 \tag{6}$$

Die Bot Success Rate ergänzt die Exception Rate, indem sie ausschließlich technische Fehler abbildet und damit die Zuverlässigkeit bewertet.

5.4.3. Metriken zum Imagegewinn

Der durch die Automatisierung entstehende Imagegewinn lässt sich nicht direkt messen, da entsprechende Lieferantenbefragungen oder Datenerhebungen den Rahmen dieser Arbeit übersteigen. Stattdessen werden zwei Ersatzgrößen herangezogen; sie dienen als Indikatoren für mögliche Reputationsvorteile, die durch eine schnellere und zuverlässigere Zahlungsabwicklung entstehen können [15, vgl. S. 2–4].

In der Literatur wird dieser Zusammenhang als Wirkungskette beschrieben [15, vgl. S. 2–4]:

(7)

Durchlaufzeit Die Durchlaufzeit misst die verstrichene Zeit zwischen dem Eingang einer Rechnung im Eingangsordner und ihrer finalen Ablage im Abschlussordner:

$$t_{\text{DLZ}} = t_{\text{Abschluss}} - t_{\text{Eingang}} \tag{8}$$

Diese Metrik ist in allen drei Implementierungen direkt über die Dateisystem-Zeitstempel messbar und dient gleichzeitig als Indikator für den Imagegewinn, da eine kürzere Verarbeitungszeit unmittelbar zu schnelleren Zahlungen und damit zu einer verbesserten Lieferantenbeziehung führt [17].

On-Time Payment Rate (theoretischer Proxy) Die On-Time Payment Rate gibt an, welcher Anteil der Lieferantenrechnungen innerhalb der vereinbarten Zahlungsfrist beglichen wird:

$$\text{On-Time Payment Rate} = \frac{\text{Pünktliche Zahlungen}}{\text{Gesamtzahlungen}} \times 100 \tag{9}$$

Da auch in diesem Fall keine realen Zahlungsdaten erhoben werden können, dient diese Kennzahl als theoretischer Referenzwert.

5.4.4. Qualitativer Plattformvergleich

Zur Beantwortung der Hauptforschungsfrage sowie der Teilforschungsfragen 2 und 3 wird ergänzend ein qualitativer Plattformvergleich durchgeführt. Dieser ist notwendig, da sich das Exception-Handling der drei betrachteten Plattformen nicht vollständig über quantitative Kennzahlen abbilden lässt. Neben messbaren Ergebnissen wie Erfolgs- oder Fehlerraten sind daher auch konzeptionelle und implementierungstechnische Unterschiede relevant, etwa die Art der Fehlerbehandlung, die Konfigurierbarkeit von Wiederholungsversuchen oder die Einbindung manueller Validierungsschritte.

Die Vergleichskriterien 1–3 sowie 5–6 orientieren sich an den offiziellen Plattformdokumentationen [45, 26, 31]. Das vierte Kriterium, das Ordner-Routing, wird im Rahmen dieser Arbeit als eigene operationale Vergleichsgröße definiert. Da keine der untersuchten Plattformen ein einheitliches Routing-Konzept vorgibt, wird betrachtet, wie Rechnungen abhängig vom Verarbeitungsergebnis in definierte Zielzustände überführt werden. Dadurch lässt sich vergleichen, wie die Plattformen ein zustandsbasiertes Fehler- und Prozessrouting praktisch unterstützen.

Für den qualitativen Plattformvergleich werden sechs Kriterien herangezogen:

1. **Geschäfts- vs. System-Handling:** Bewertet wird, ob die Plattform fachliche Ausnahmen, beispielsweise Betragsabweichungen, von technischen Fehlern, beispielsweise Verbindungs- oder Serverproblemen, eindeutig unterscheiden kann.

2. **Retry-Konfigurierbarkeit:** Untersucht wird, in welchem Umfang Wiederholungsversuche bei vorübergehenden Fehlern konfiguriert werden können, etwa hinsichtlich der Anzahl der Versuche oder der Wartezeit zwischen den Ausführungen.

3. **Workflow-Handling (Try/Catch):** Betrachtet wird, ob strukturierte Mechanismen zur Fehlerbehandlung vorhanden sind und wie aufwendig deren Umsetzung auf Aktivitäts- oder Codeebene ist.

4. **Ordner-Routing:** Geprüft wird, wie Rechnungen abhängig vom Ergebnis der Verarbeitung in definierte Zielordner verschoben und dadurch unterschiedlichen Prozesszuständen zugeordnet werden können.

5. **HITL-Validierungsstation:** Bewertet wird, wie Fälle mit unsicheren oder fehlerhaften Ergebnissen an eine manuelle Validierung übergeben werden und welcher Implementierungsaufwand damit verbunden ist.

6. **Logging & Observability:** Untersucht wird, wie detailliert Fehler, Ausnahmen und Prozesszustände protokolliert werden und wie gut diese Informationen im Betrieb nachvollziehbar sind.

5.4.5. Übersicht der ausgewählten Metriken

Tabelle 5 fasst alle ausgewählten Metriken zusammen und ordnet sie den jeweiligen Forschungsfragen zu.

Tabelle 5: Übersicht der ausgewählten Metriken und ihre Zuordnung zu den Forschungsfragen.

Nr.	Metrik	Forschungsfrage
1	ROI	TF4
2	Amortisationsdauer	TF4
3	FTE-Einsparung	TF4
4	STP-Rate	TF5
5	Exception Rate (nach Typ)	TF3, TF5, HF
6	Bot Success Rate	TF5, HF
7	Durchlaufzeit	TF4, TF5
8	On-Time Payment Rate (Proxy)	TF4
9	Qualitativer Plattformvergleich	HF, TF2, TF3

HF = Hauptforschungsfrage; TF = Teilforschungsfrage

5.5. Testaufbau

Dieser Abschnitt beschreibt das methodische Vorgehen, um die drei implementierten RPA-Lösungen unter gleichen Bedingungen zu testen und zu vergleichen. Ziel des Versuchsaufbaus ist es, die Performanz, die Fehlertoleranz und die Wirtschaftlichkeit der Plattformen anhand eines standardisierten Testverfahrens quantitativ und qualitativ zu evaluieren.

5.5.1. Testumgebung und Systemkonfiguration

Um externe Störfaktoren und Netzwerklatenzen auszuschließen, werden alle Testläufe lokal auf einer homogenen Systemumgebung ausgeführt. Als Ziel-ERP-System dient eine lokale Instanz von Dolibarr, die über eine XAMPP-Umgebung (Apache-Webserver und MySQL-Datenbank) unter Windows 11 betrieben wird. Tabelle 6 fasst die technischen Komponenten, Softwareversionen sowie die einheitlich definierte Konfidenzschwelle für die KI-gestützte Extraktion zusammen.

Tabelle 6: Technische Komponenten und Konfiguration der Testumgebung.

Komponente	Spezifikation / Konfiguration
Betriebssystem	Windows 11
ERP-System	Dolibarr (lokal über XAMPP: Apache, MySQL)
UiPath	Studio 2024.10.8 (mit Document Understanding)
Power Automate	Desktop 2.68.237.26118 & Power Platform Cloud (AI Builder)
Sema4.ai / Robocorp	Python 3.10.1 mit `rpaframework` 32.0.0
KI-Extraktion	UiPath Document Understanding / AI Builder / Nanonets OCR API v2
Konfidenzschwelle	80 % (einheitlich über alle drei Plattformen konfiguriert)

5.5.2. Struktur des Testkorpus und vordefinierte Fehlerszenarien

Die empirische Evaluation basiert auf einem standardisierten Testkorpus, der aus einer festlegten Gesamtzahl an PDF-Rechnungen besteht. Für jedes dieser Dokumente wird vorab ein eindeutiger theoretischer Soll-Zustand (Ground Truth) definiert. Dieser methodische Ansatz ermöglicht es, nach Durchführung der Testläufe nicht nur die reine quantitative Verarbeitungsquote zu erfassen, sondern die funktionale Korrektheit des prozessualen Routings explizit zu überprüfen.

Der Testaufbau differenziert sich in folgende funktionale Szenarien:

- **Standardverarbeitung (Happy Path):** Formell und rechnerisch fehlerfreie Belege, bei denen sämtliche extrahierten Werte die definierte Konfidenzschwelle überschreiten. Dieses Szenario dient als Baseline zur Messung der autonomen Dunkelbuchungsfähigkeit der Systeme.
- **Inhaltliche Diskrepanzen (Business Exceptions):** Belege mit gezielt eingebauten rechnerischen Abweichungen, wie beispielsweise einer mathematischen Ungleichheit zwischen der summierten Positionsebene und dem ausgewiesenen Bruttobetrag. Dieses Szenario überprüft die Fähigkeit der Plattformen, logische Geschäftsfehler autonom zu isolieren.
- **Qualitative Erkennungsmängel (Human-in-the-Loop):** Dokumente, die durch künstliche Bildunschärfe oder fehlerhafte Formatierungen eine Unterschreitung der Konfidenzschwelle erzwingen. Damit wird die nahtlose Übergabe an die jeweilige manuelle Validierungsstation sowie die anschließende Fortsetzungsfähigkeit des Bots getestet.
- **Infrastrukturelle Instabilitäten (System Exceptions & Auto-Retry):** Szenarien, die nicht primär auf dem Dokumentenmaterial basieren, sondern bei denen zur Laufzeit technische Systemfehler provoziert werden (z. B. durch die temporäre Deaktivierung des ERP-Serverdienstes). Hiermit wird evaluiert, wie effektiv die Plattformen transiente Fehler über zeitlich parametrisierte Wiederholungslogiken (Retries) abfangen oder kontrollierte Fallback-Routinen einleiten.

5.5.3. Versuchsdurchführung und Datenerhebung

Um eine strikte Vergleichbarkeit zu gewährleisten, durchläuft jede der drei RPA-Plattformen denselben Testkorpus unter identischen Rahmenbedingungen. Der Prozess steuert seinen Zustand hierbei dynamisch über ein definiertes Ordnersystem: Jedes Dokument wandert während der Verarbeitung durch spezifische Verzeichnisse, die den aktuellen Bearbeitungsstand repräsentieren.

Die im Rahmen der HITL-Schleifen notwendigen manuellen Korrekturen und Datennacheingaben werden durchgehend von derselben Person durchgeführt. Dadurch wird der Faktor des menschlichen Bedienereinflusses als Störvariable über alle Plattformen hinweg konstant gehalten.

5.5.4. Messung und Zuordnung der Evaluationsmetriken

Die Datengewinnung erfolgt primär über die anschließende quantitative Auswertung der Ordnerstrukturen nach Beendigung der Testläufe. Ein automatisiertes Python-Skript erfasst die Verteilung der Dokumente auf die Endordner und berechnet daraus Kernmetriken wie die STP-Rate, die differenzierte Exception Rate sowie die allgemeine Bot Success Rate.

Tabelle 7 systematisiert die untersuchten Metriken nach ihrem Status (gemessen, qualitativ bewertet oder über vordefinierte mathematische Modelle geschätzt).

Tabelle 7: Systematisierung und Erhebungswege der Evaluationsmetriken.

Untersuchte Metrik	**Methodischer Erhebungsweg**
STP-Rate	Quantitativ über finale Ordnerzählung (Ziel 003)
Exception Rate (nach Typ)	Quantitativ über Fehlerordner (007 / 008 / 009)
Bot Success Rate	Verhältnis erfolgreicher Läufe zu Systemabbrüchen
Durchlaufzeit	Protokollierung über Datei-Zeitstempel (inkl. HITL-Wartezeiten)
Qualitativer Vergleich	Kriterienbasierte Architekturanalyse der Implementierungen
ROI / Amortisation	Mathematische Modellierung anhand von Kosten-Nutzen-Formeln
FTE-Einsparung	Projektion basierend auf gemessenen Bearbeitungszeiten
On-Time Payment Rate	Literaturgestützte Ableitung (keine realen Bankdaten)

Die für die Wirtschaftlichkeits- und Aufwandsschätzungen notwendigen Variablen, wie etwa der jeweilige Implementierungs- und Entwicklungsaufwand, werden separat erfasst. Hierzu werden die Entwicklungszeiten, Umfang, die strukturelle Komplexität der Workflows gemessen sowie die jeweiligen Fehler- und Validierungslogiken gegenübergestellt.

6. Implementierung der Prototypen

Dieses Kapitel beschreibt, wie der in Kapitel 5 vorgestellte Prozess auf den drei Plattformen tatsächlich umgesetzt wurde. Für jede Plattform werden vier Aspekte betrachtet: der Standardprozess (der reguläre Ablauf ohne Störung), die Umsetzung des Exception-Handlings, die manuelle Validierung im Human-in-the-Loop und schließlich Logging und Fehlerablage. Diese Aufteilung greift die Teilforschungsfragen nach den Implementierungsschritten und nach dem Aufwand für das Exception-Handling auf.

6.1. Gemeinsame Anforderungen an alle Prototypen

Damit die drei Umsetzungen vergleichbar bleiben, müssen sie dieselben Anforderungen erfüllen. Alle lesen dieselben Kopf- und Positionsfelder aus (siehe Tabelle 2) und buchen gegen dieselbe Dolibarr-Instanz. Alle verwenden dieselbe Konfidenzschwelle von 80 %, ab der ein Feld als unsicher gilt und die manuelle Validierung ausgelöst wird. Alle ordnen jede Rechnung am Ende genau einem der vier Endzustände 003, 007, 008 und 009 zu und steuern diesen Zustand über dasselbe Ordnersystem (siehe Tabelle 3). Und alle sichern vorübergehende Störungen über einen Wiederholungsmechanismus ab, statt sofort abzubrechen. Die Prototypen unterscheiden sich damit nicht im fachlichen Verhalten, sondern allein in der technischen Umsetzung dieser Anforderungen.

6.2. Implementierung mit UiPath und Document Understanding

6.2.1. Standardprozess

Der Prototyp entsteht im UiPath Studio und ist überwiegend per Drag-and-Drop aus Aktivitäten zusammengesetzt. Die Hauptdatei `Main.xaml` bildet den Rahmen: Sie liest die Dateiliste ein, meldet sich über einen eigenen Workflow am ERP an und arbeitet die Rechnungen in einer Schleife ab, deren Bedingung `main_aktuellerIndex < main_dateiListe.Count` lautet. Je Rechnung wird die Datei nach 002 verschoben und an Document Understanding übergeben, das die Felder mitsamt Konfidenzwerten erkennt. Insgesamt umfasst die Lösung rund 96 Aktivitäten, verteilt auf fünf Workflows. Abbildung 7 zeigt das Grundgerüst und die Extraktionsschritte. Im Anhang A befindet sich der gesamte Workflow.

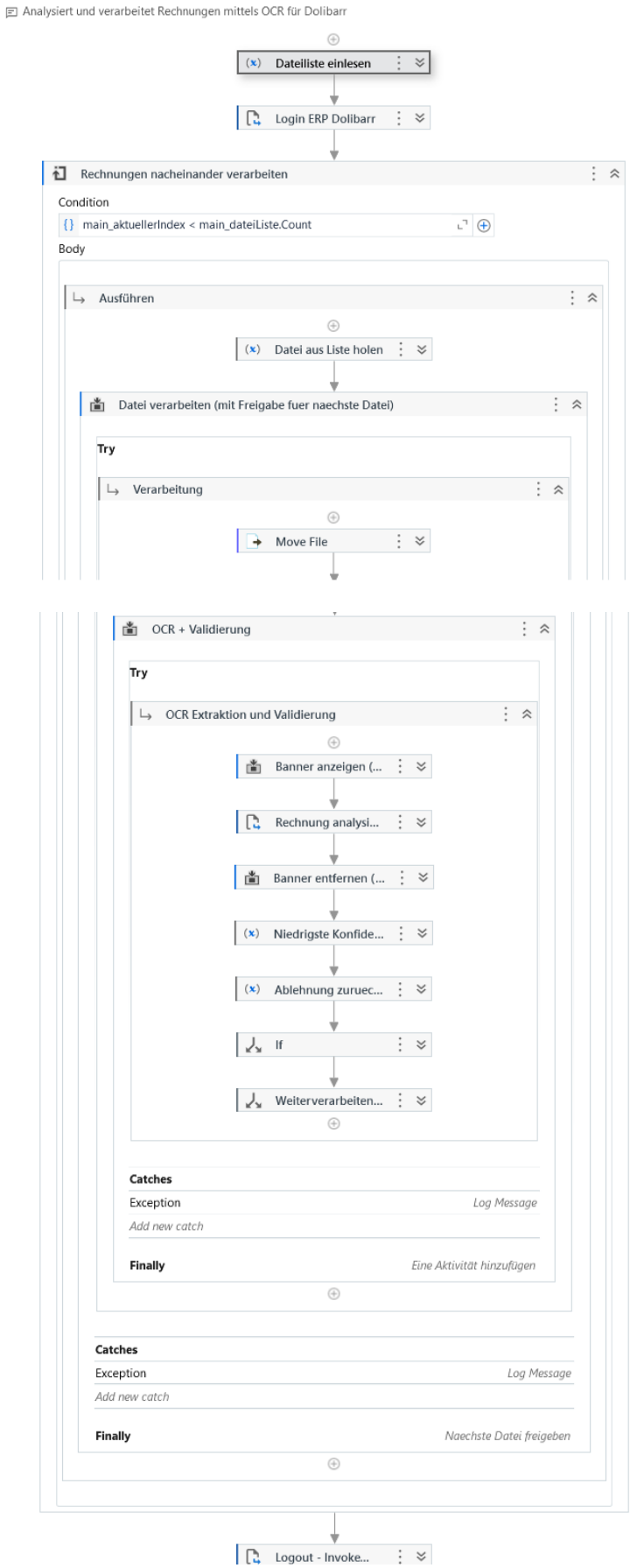


Abbildung 7: Standardprozess in UiPath: Grundgerüst (oben) und Extraktionsschritte mit äußerer Fehlerbehandlung (unten).

6.2.2. Umsetzung des Exception-Handlings

UiPath behandelt Fehler deklarativ über verschachtelte Try-Catch-Aktivitäten. Im Prototyp liegen mehrere solcher Blöcke ineinander: Ein äußerer Block (`exceptionGuard`) umschließt die gesamte Verarbeitung einer Rechnung und sorgt im *Finally*-Zweig dafür, dass über „Nächste Datei freigeben“ stets zur nächsten Rechnung übergegangen wird. Darunter fangen weitere Blöcke gezielt Teilfehler ab, etwa die Ablehnung in der Validierung (`exReject`) oder Fehler beim Entfernen des Statusbanners (`exBanner`). Vorübergehende Störungen werden über einen *RetryScope* mehrfach wiederholt. Ein technischer Fehler wird so kontrolliert aufgefangen und führt über eine *Move-File*-Aktivität zur Ablage in 009, während die Schleife weiterläuft. Abbildung 8 zeigt die verschachtelte Struktur.

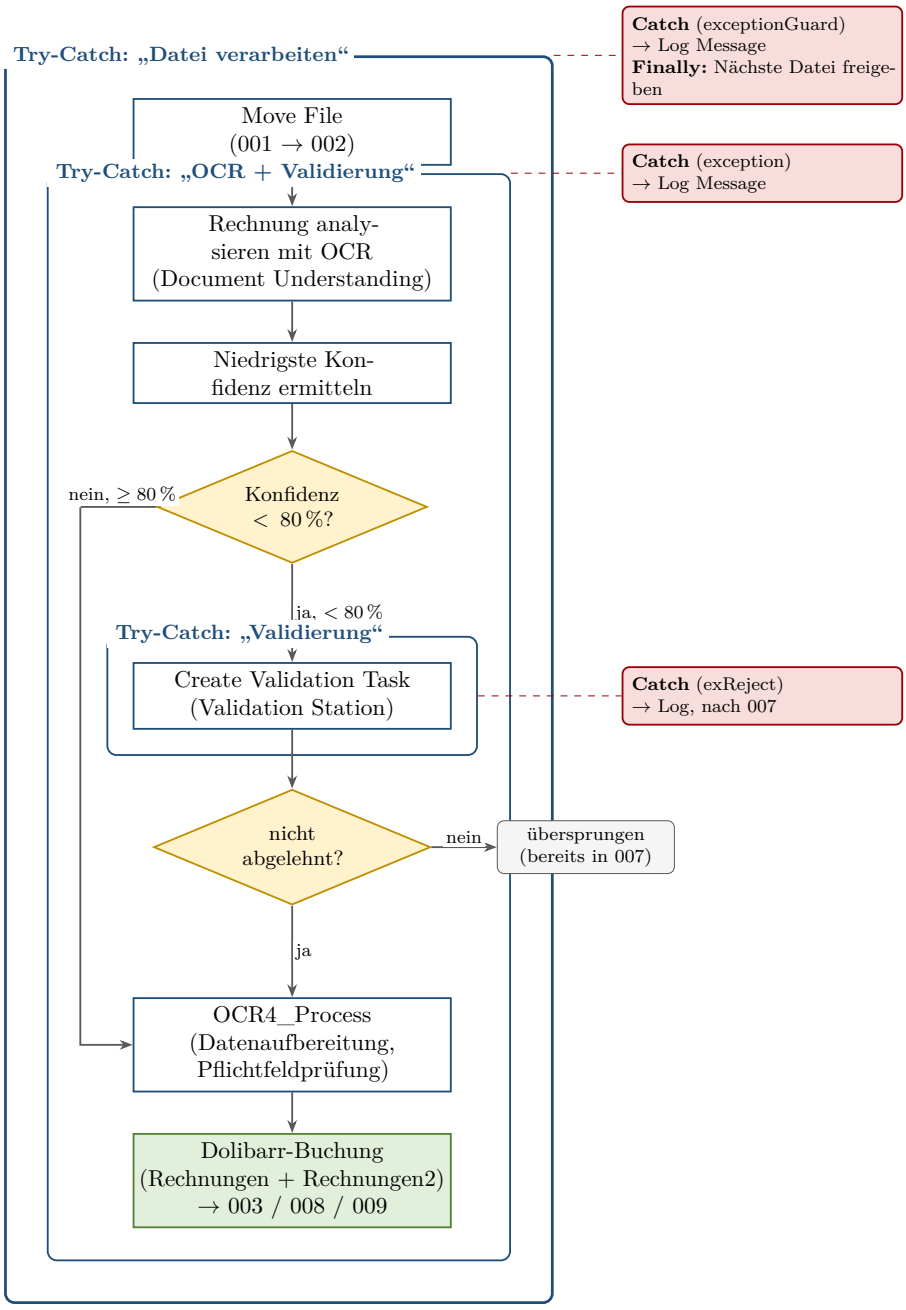


Abbildung 8: Verschachtelte Try-Catch-Struktur des UiPath-Prototyps (aus `Main.xaml`).

6.2.3. Umsetzung der manuellen Validierung

Die manuelle Prüfung nutzt die in Document Understanding integrierte Validation Station (siehe Abbildung 9). Eine *If*-Aktivität prüft, ob die niedrigste Feldkonfidenz unter der Schwelle liegt (`main_minKonfidenz < 80`). Trifft das zu, öffnet der *Then*-Zweig über „Action Center öffnen“ und „Create Validation“ die Validierungsoberfläche, in welcher der Bearbeiter die Felder korrigiert oder die Rechnung ablehnt. Eine Ablehnung wird als Geschäftsfehler behandelt und über den `exReject`-Zweig nach 007 verschoben. Liegt keine Unsicherheit vor, überspringt der *Else*-Zweig die Validierung und setzt die Verarbeitung fort. Der Implementierungsaufwand ist hier gering, weil die Oberfläche fertig mitgeliefert wird.

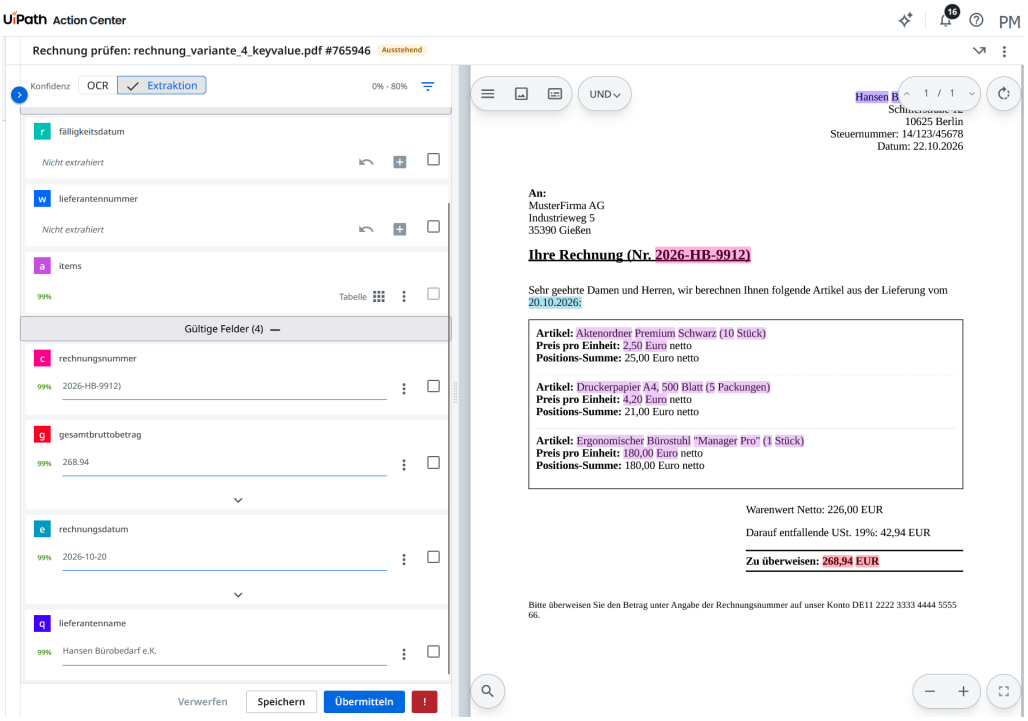


Abbildung 9: Manuelle Validierung in UiPath über die integrierte Validation Station.

6.2.4. Logging und Fehlerablage

UiPath bringt mit der *Log-Message*-Aktivität ein strukturiertes Logging mit, das Meldungen in die Ausführungsprotokolle schreibt. Der Prototyp nutzt sie an den Fehlerstellen, etwa in den Catch-Zweigen. Die Fehlerablage selbst geschieht über *Move-File*-Aktivitäten, die die Rechnung je nach Ergebnis in den passenden Statusordner verschieben. Logging und Routing sind damit eigenständige, deklarative Aktivitäten und nicht im Anwendungscode verstreut.

6.3. Implementierung mit Microsoft Power Automate und AI Builder

Die Power-Automate-Lösung ist hybrid aufgebaut. Ein Cloud-Flow übernimmt die Orchestrierung, während die eigentliche Bedienung von Dolibarr in Desktop-Flows (Power Automate Desktop) abläuft. Der Cloud-Flow startet manuell und meldet sich zunächst einmalig über einen Login-Desktop-Flow an Dolibarr an. Anschließend listet er die Dateien im Eingangsordner und prüft über eine Bedingung, ob überhaupt Rechnungen vorliegen. Ist der Ordner leer, meldet sich der Flow direkt wieder ab und endet; andernfalls arbeitet er die Rechnungen in einer *For-each*-Schleife ab und meldet sich erst nach der letzten Rechnung wieder ab. Die Extraktion übernimmt der AI Builder mit dem vorgefertigten Modell zur Rechnungsverarbeitung. Insgesamt besteht der Cloud-Flow aus 15 Aktionen; die Desktop-Flows umfassen zusammen rund 180 Aktionen. Abbildung 10 zeigt den vollständigen Ablauf und ordnet jeden Schritt einer der drei Schichten zu.

Die Anmeldung liegt bewusst außerhalb der Schleife. Da der Cloud-Flow die Bedienung von Dolibarr je Rechnung an einen Desktop-Flow übergibt, hätte ein Login innerhalb der *For-each*-Schleife bei jedem Durchlauf eine erneute Anmeldung ausgelöst. Die Sitzung wird daher einmal geöffnet und über alle Rechnungen offen gehalten. Die beiden anderen Prototypen melden sich ebenfalls nur einmal an, sodass das Laufzeitverhalten vergleichbar bleibt; in Power Automate war dafür allerdings eine eigene strukturelle Anpassung nötig, die der hybride Aufbau aus Cloud- und Desktop-Flow geschuldet ist.

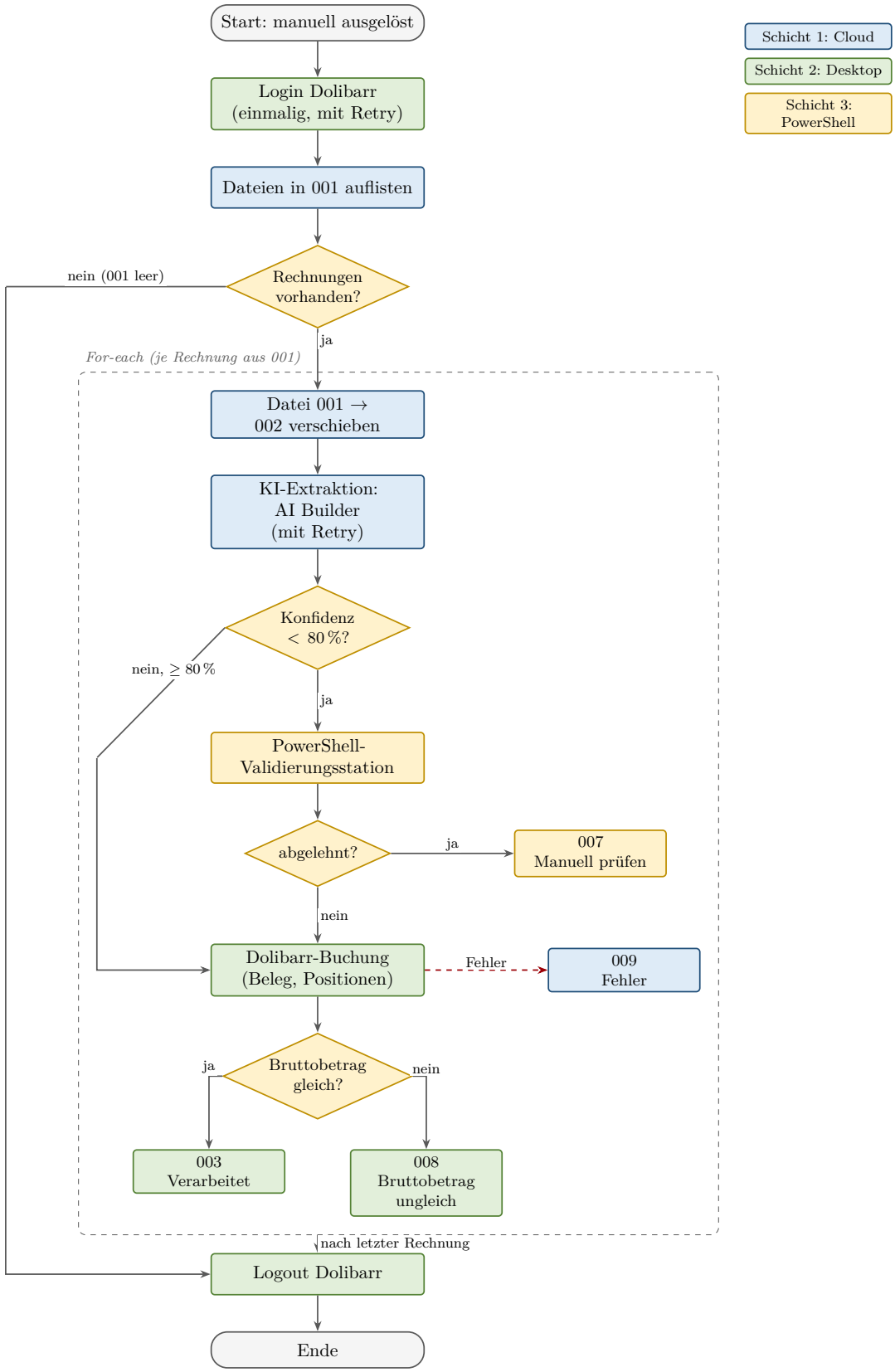


Abbildung 10: Ablauf des Power-Automate-Prototyps mit Zuordnung der Schritte zu den drei Schichten (Cloud, Desktop, PowerShell).

6.3.1. Umsetzung des Exception-Handlings

Power Automate kennt keine eigene Try-Catch-Aktivität. Stattdessen wird die Fehlerbehandlung über zwei Mechanismen abgebildet. Vorübergehende Störungen fängt eine `retryPolicy` ab, die einzelne Aktionen mehrfach wiederholt; im Prototyp ist sie etwa beim Login und bei der Extraktion auf drei Versuche mit einer Minute Abstand eingestellt. Echte Fehler werden über die `runAfter`-Einstellung geroutet: Eine nachgelagerte Aktion läuft gezielt nur dann, wenn die vorherige fehlgeschlagen, übersprungen oder in einen Timeout gelaufen ist, und verschiebt die Rechnung dann nach 009. Listing 1 zeigt beide Mechanismen.

```
"Prozessdokumente": {
    "type": "OpenApiConnection",
    "inputs": { "...": "AI-Builder-Aufruf",
        "retryPolicy": { "type": "fixed", "count": 3, "interval": "PT1M" } }
},
"Datei_verschieben_nach_Fehlerordner": {
    "type": "OpenApiConnection",
    "inputs": { "destination": "/Rechnungen/009 - Fehler/..." },
    "runAfter": { "Prozessdokumente": [ "Failed", "Skipped", "TimedOut" ] }
}
```

Listing 1: Power Automate – Retry über `retryPolicy` und Fehler-Routing über `runAfter` (Auszug aus der Flow-Definition).

6.3.2. Umsetzung der manuellen Validierung

Die manuelle Validierung war bei Power Automate der aufwendigste Teil. Die im AI Builder vorgesehene Validierungsoberfläche ließ sich im verfügbaren Umfeld nicht praktikabel einsetzen: Lizenz-, Administrator-, Mandanten- und Zugriffsrechte standen im Weg, Teile der Dokumentation waren fehlerhaft, und der Versuch, Flows und das Extraktionsmodell über einen separaten Account zu importieren und zu exportieren, scheiterte an weiteren Hürden. Aus diesem Grund wurde eine eigene Validierungsstation entwickelt: ein PowerShell-Skript mit WinForms-Oberfläche von rund 435 Zeilen, das aus dem Desktop-Flow über die Aktion „PowerShell-Skript ausführen“ aufgerufen wird. Es zeigt die extrahierten Daten, hebt Felder unter 80 % farblich hervor und bietet zwei Schaltflächen: „Validieren & in Excel speichern“ und „Ablehnen → 007“. Abbildung 11 zeigt die laufende Oberfläche, Listing 2 zeigt den Kern.

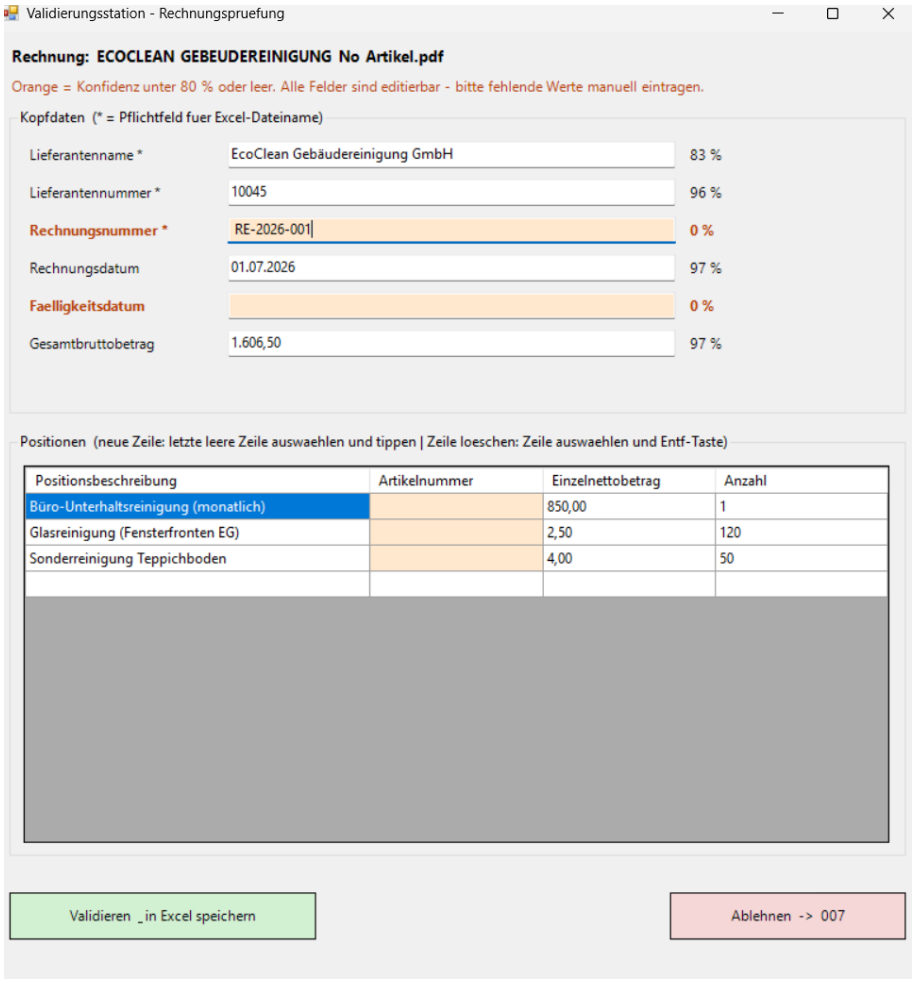


Abbildung 11: Selbstgebaute Validierungsstation in Power Automate (PowerShell/WinForms).

```powershell
param([string]$InputJson = "", [double]$Schwellwert = 0.80)

# Feld unsicher? -> in der Oberflaeche orange markieren
function Test-KonfidenzNiedrig {
    param($Konfidenz)
    return ([double]$Konfidenz -lt $Schwellwert)
}

# Button "Validieren & in Excel speichern" -> status = validated
$btnValidieren.Add_Click({
    # ... korrigierte Werte einsammeln und in die Excel-Datei schreiben ...
    $form.Tag = "validated"; $form.Close()
})

# Button "Ablehnen -> 007" -> Datei verschieben, status = rejected
$btnAblehnen.Add_Click({
    if ($antwort -eq "Yes") {
        Move-Item $quellpfad $daten.pfade.ordner_007
        $form.Tag = "rejected"; $form.Close()
    }
})
```

Listing 2: Power Automate – Kern der selbstgebauten Validierungsstation (PowerShell/WinForms): Schwellwertprüfung und die zwei Schaltflächen.

6.3.3. Logging und Fehlerablage

Beim Logging spielt Power Automate seinen Cloud-Charakter aus: Der Laufverlauf protokolliert jede Aktion samt Status automatisch, ohne dass dafür eigene Schritte nötig sind. Ergänzend setzt der Prototyp Log-Message-Aktionen ein. Die Fehlerablage erfolgt über Move-File-Aktionen, die die Rechnung nach 003 bzw. 009 verschieben; die Entscheidung zwischen den Zuständen trifft eine If -Aktion anhand des Status, den der Desktop-Flow zurückliefert. Abbildung 12, sowie Listing 3 und Listing 4 zeigen den Prozessabbruch.

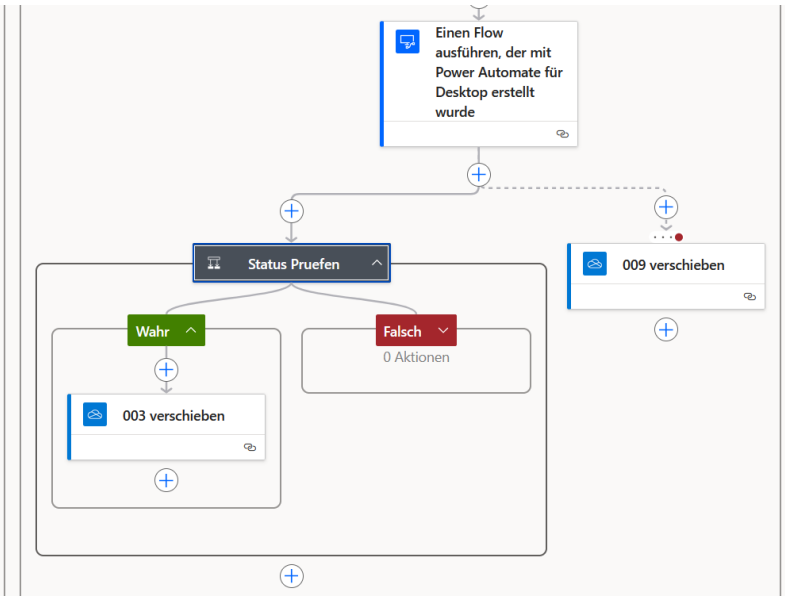


Abbildung 12: Logging in Power Automate über den automatischen Cloud-Laufverlauf.

```
"runAfter": {
  "Einen_Flow_ausführen,_der_mit_Power_Automate_für_Desktop_erstellt_wurde": [
    "SUCCEEDED"
  ]
}
```

Listing 3: Power Automate – Verschiebung nach 003, nach erfolgreichen Durchlauf des Power Automate Desktop Flows.

```
"runAfter": {
  "Einen_Flow_ausführen,_der_mit_Power_Automate_für_Desktop_erstellt_wurde": [
    "Failed"
  ]
}
```

Listing 4: Power Automate – Verschiebung nach 009, nach gescheiterten Durchlauf des Power Automate Desktop Flows.

6.4. Implementierung mit Sema4.ai und Nanonets

6.4.1. Standardprozess

Die dritte Umsetzung ist vollständig in Python programmiert und baut auf dem Robocorp-Framework auf. Den Rahmen bildet die mit `@task` markierte Funktion `main_invoice_pipeline`. Sie sucht die unbearbeiteten Rechnungen, meldet sich einmalig an Dolibarr an und arbeitet

die Dateien in einer Schleife ab. Die Bedienung von Dolibarr läuft über eine Playwright-gestützte Browsersteuerung, die Extraktion über die REST-API von Nanonets. Abbildung 13 zeigt zunächst die Architektur im Überblick. Der gesamte Ablauf liegt damit offen im Code (rund 665 Zeilen) und ist anschließend in Listing 5 in gekürzter Form zu sehen.

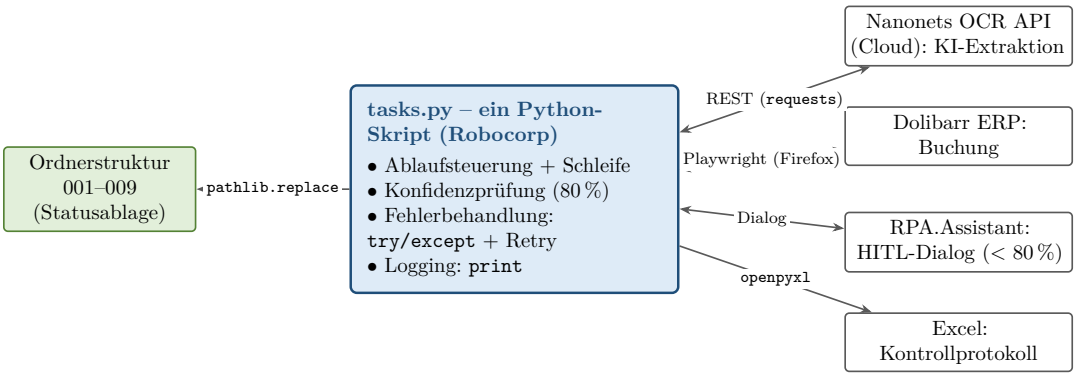


Abbildung 13: Architektur des Sema4.ai-Prototyps und Anbindung der externen Schnittstellen (Nanonets, Dolibarr, Excel).

```python
@task
def main_invoice_pipeline():
    rechnungs_dateien = list(ORDNER_001_Unbearbeitet.glob("*.pdf"))
    if not rechnungs_dateien:
        return
    login_dolibarr_mit_retry()                    # einmaliger Login (mit Retry)

    for datei in rechnungs_dateien:
        ziel_pfad = ORDNER_002_In_Verarbeitung / datei.name
        datei.replace(ziel_pfad)                  # 001 -> 002
        daten = extraktion_nanonets(ziel_pfad)    # KI-Extraktion (Nanonets)

        if daten:
            if braucht_validierung(daten):        # mind. ein Feld < 80 %?
                aktion, ergebnis = nanonets_review_station(daten)   # HITL
                if aktion == "abbrechen":
                    verschiebe_nach_007_mit_begruendung(ziel_pfad, ergebnis)
                    continue
                daten = ergebnis                  # korrigierte Werte uebernehmen
            try:
                speichere_in_excel(daten, excel_pfad)
                erfolg = rechnung_erstellen_dolibarr(daten)
                ziel_pfad.replace(ORDNER_003_Verarbeitet / datei.name if erfolg
                                  else ORDNER_008_Bruttobetrag_ungleich / datei.
                                      name)
            except Exception as fehler:           # Systemfehler abfangen
                ziel_pfad.replace(ORDNER_009_Fehler / datei.name)
        else:
            ziel_pfad.replace(ORDNER_009_Fehler / datei.name)  # Extraktion
                fehlgeschlagen

    logout_dolibarr()
```

Listing 5: Sema4.ai – Hauptablauf der Pipeline (gekürzt): Statuswechsel 001 → 002, KI-Extraktion, optionale Validierung, Buchung und Ablage in 003/008/009.

6.4.2. Umsetzung des Exception-Handlings

Anders als bei den Low-Code-Werkzeugen wird die Fehlerbehandlung hier vollständig ausprogrammiert. Die Buchung steht in einem `try/except`-Block; jeder unerwartete Laufzeitfehler wird abgefangen und führt zur Ablage in 009. Vorübergehende Störungen deckt eine selbst geschriebene Retry-Schleife ab, die zwei Fehlerklassen unterscheidet: Verbindungsprobleme und Server-Fehler (5xx) werden wiederholt, ein echter API-Fehler (4xx, etwa ein falscher Schlüssel) bricht dagegen sofort ab, weil eine Wiederholung zwecklos wäre. Listing 6 zeigt diese Logik. Auch der Login ist auf dieselbe Weise abgesichert: Er prüft die Erreichbarkeit mehrfach und bricht erst nach erfolglosen Versuchen kontrolliert ab.

```
for versuch in range(1, NANONETS_RETRY_MAX + 1):
    try:
        antwort = requests.post(url, auth=(NANONETS_API_KEY, ""),
                                files={"file": datei_objekt}, timeout=120)
        if antwort.status_code == 200:
            break                                   # Erfolg -> Schleife verlassen
        if antwort.status_code >= 500:
            print(f"   Server-Fehler {antwort.status_code} (Versuch {versuch})")
                 # 5xx -> Retry
        else:
            return None                             # 4xx (falscher Key/Modell) ->
                KEIN Retry
    except (requests.exceptions.ConnectionError,
            requests.exceptions.Timeout) as err:
        antwort = None                              # Verbindungsproblem -> Retry
    if versuch < NANONETS_RETRY_MAX:
        time.sleep(NANONETS_RETRY_WARTE_SEK)        # warten, dann erneut
```

Listing 6: Sema4.ai – Retry mit Unterscheidung der Fehlerklassen: 5xx und Verbindungsfehler werden wiederholt, 4xx bricht sofort ab.

6.4.3. Umsetzung der manuellen Validierung

Die Validierung ist ebenfalls im Code umgesetzt. Die Funktion `braucht_validierung` prüft, ob ein Kopffeld oder eine Tabellenzelle unter der Schwelle liegt (Listing 7). Ist das der Fall, öffnet `nanonets_review_station` einen Dialog über `RPA.Assistant` und pausiert die Verarbeitung (Listing 8). Der Bearbeiter kann die Werte in der Nanonets-Oberfläche korrigieren und freigeben; der Prozess ruft die korrigierten Daten danach erneut ab. Bricht der Bearbeiter ab, wird die Rechnung nach 007 verschoben. Im Unterschied zu UiPath gibt es hier keine fertige Validierungsoberfläche – die Logik und der Dialog sind selbst geschrieben, fügen sich aber nahtlos in den Code ein.

```
def braucht_validierung(daten, schwelle=KONFIDENZ_SCHWELLE):   #
    KONFIDENZ_SCHWELLE = 0.80
    konfidenz = daten.get("_konfidenz", {})
    for daten_key, _, _ in VALIDIERUNGS_FELDER:               # Kopf-Attribute
        pruefen
        if konfidenz.get(daten_key, 0.0) < schwelle:
            return True
    for zeile_konfidenz in daten.get("_tabellen_konfidenz", []):  #
        Tabellenzellen pruefen
        for score in zeile_konfidenz.values():
            if score < schwelle:
                return True
    return False
```

Listing 7: Sema4.ai – Konfidenzprüfung: sobald ein Feld oder eine Zelle unter 80 % liegt, wird die Validierung ausgelöst.

```
assistant = Assistant()
assistant.add_heading("Nanonets Review erforderlich")
assistant.add_text("Folgende Felder liegen unter 80 % Konfidenz:\n" +
    zusammenfassung)
assistant.add_submit_buttons(buttons="Weiter (akt. Daten abrufen),Abbrechen",
                             default="Weiter (akt. Daten abrufen)")
ergebnis = assistant.run_dialog(timeout=1800, on_top=True,
                                title="Nanonets Review - Warte auf Bestaetigung")

if getattr(ergebnis, "submit", "").startswith("Abbrechen"):
    return ("abbrechen", begruendung)             # -> Ordner 007
aktualisiert = extraktion_nanonets_by_file_id(file_id)   # korrigierte Werte
    abrufen
return ("weiter", aktualisiert)
```

Listing 8: Sema4.ai – Human-in-the-Loop-Dialog über `RPA.Assistant`; "Abbrechen" verschiebt nach 007, "Weiter" ruft die korrigierten Werte erneut ab.

6.4.4. Logging und Fehlerablage

Das Logging ist bei Sema4.ai bewusst schlank gehalten und besteht aus `print`-Ausgaben auf der Konsole. Ein zentrales Protokoll wie bei UiPath oder der Cloud-Laufverlauf von Power Automate gibt es nicht. Die Fehlerablage erfolgt über `pathlib.replace`, das die Rechnung in den passenden Statusordner verschiebt. Listing 9 zeigt beides am Beispiel des Bruttobetrags-Vergleichs.

```
if rechnung_eintragung_erfolgreich:
    ziel_pfad.replace(ORDNER_003_Verarbeitet / datei.name)
    print(f"   [ERFOLG] Betraege gleich. Datei nach '003 - Verarbeitet'
        verschoben.")
else:
    ziel_pfad.replace(ORDNER_008_Bruttobetrag_ungleich / datei.name)
    print(f"   [WARNUNG] Betraege ungleich! Datei nach '008 - Bruttobetrag
        ungleich' verschoben.")
```

Listing 9: Sema4.ai – Logging über `print` und Ergebnis-Routing über `pathlib.replace`.

7. Ergebnisse

Dieses Kapitel stellt die Messergebnisse der drei Prototypen dar. Alle drei Lösungen verarbeiteten denselben Satz von 30 Rechnungen; lediglich die Reihenfolge der Abarbeitungen unterscheiden sich. Die Prozesseffizienz-Kennzahlen stammen aus dem Skript `auswertung_ordner.py`, das die PDF-Dateien in den Zielordnern 003, 007, 008 und 009 auszählt und daraus die Straight-Through-, die Exception- und die Bot-Success-Rate berechnet. Die Durchlaufzeiten stammen aus dem Skript `dlz_watcher.py`, das während des Laufs mitläuft und den Zeitpunkt festhält, zu dem jede Rechnung in einem Abschlussordner erscheint. Im Anhang B befinden sich die beiden Dateien. Das Verhalten in den Ausnahmefällen wurde während der Läufe beobachtet. Die Darstellung bleibt beschreibend; eine Einordnung und Bewertung erfolgt im Kapitel 8.

7.1. Zielablage der Rechnungen

Tabelle 8 zeigt, in welchen Endordnern die 30 Rechnungen je Plattform abgelegt wurden. Auffällig ist die unterschiedliche Belegung von Ordner 008. Obwohl alle Plattformen dieselben Rechnungen erhielten, stuften sie verschieden viele als Betragsabweichung ein: UiPath zwei, Sema4.ai sieben und Power Automate sechzehn. Ordner 007 blieb durchgängig leer; in keinem Lauf wurde eine Rechnung in der manuellen Validierung abgelehnt, alle unsicheren Fälle wurden korrigiert und freigegeben. Ein technischer Fehler (009) trat nur bei Power Automate auf und zwar ein einziges Mal.

Tabelle 8: Verteilung der 30 Rechnungen auf die Endordner je Plattform.

Zielordner	**UiPath**	**Power Automate**	**Sema4.ai**
003 – Verarbeitet	28	13	23
007 – Manuell ablehnen	0	0	0
008 – Betragsabweichung	2	16	7
009 – Fehler	0	1	0
Summe	30	30	30

7.2. Prozesseffizienz

Aus der Ordnerbelegung ergeben sich die Prozesseffizienz-Metriken in Tabelle 9. Die STP-Rate misst hier den Anteil der in Ordner 003 abgelegten Rechnungen. UiPath erreicht mit 93,3 % den höchsten Wert, gefolgt von Sema4.ai mit 76,7 % und Power Automate mit 43,3 %. Die Exception Rate verhält sich spiegelbildlich und besteht fast vollständig aus Geschäftsfehlern, also Rechnungen in Ordner 008. Die Bot Success Rate, die ausschließlich technische Abbrüche erfasst, liegt bei UiPath und Sema4.ai bei 100 % und bei Power Automate aufgrund des einen Abbruchs bei 96,7 %.

Tabelle 9: Prozesseffizienz-Metriken je Plattform (Anteile in Prozent).

Metrik	**UiPath**	**Power Automate**	**Sema4.ai**
STP-Rate (Ablage in 003)	93,3	43,3	76,7
Exception Rate	6,7	56,7	23,3
davon Geschäftsfehler (007/008)	6,7	53,3	23,3
davon Systemfehler (009)	0,0	3,3	0,0
Bot Success Rate	100,0	96,7	100,0

7.3. Durchlaufzeit

Tabelle 10 fasst die Durchlaufzeiten zusammen. UiPath verarbeitete den Stapel mit Abstand am schnellsten, Power Automate benötigte fast die doppelte Gesamtzeit. Der Median liegt bei UiPath bei 39 Sekunden je Rechnung, bei Sema4.ai bei 56 Sekunden und bei Power Automate bei 70 Sekunden. Diese Werte sind als grober Indikator zu lesen, da sie neben der reinen Bot-Bearbeitung auch Wartezeiten enthalten. Die längsten Einzelzeiten von über drei Minuten traten vereinzelt auf und heben den Mittelwert an; der Median ist daher das robustere Maß.

Tabelle 10: Durchlaufzeiten je Plattform (erfasst mit `dlz_watcher.py`).

Plattform	**Gesamtlaufzeit**	**Ø je Rechnung**	**Median**	**längste**
UiPath	25,6 min	51,2 s	39,0 s	220,2 s
Power Automate	49,8 min	99,7 s	70,1 s	228,2 s
Sema4.ai	32,7 min	65,4 s	56,1 s	160,1 s

Bei Power Automate enthält die Aufzeichnung zwei Einträge mit einer Einzelzeit von null Sekunden. Sie entstehen, wenn zwei Rechnungen innerhalb desselben Messintervalls des Mitschreibers im Abschlussordner erscheinen, und sind keine real gemessenen Durchlaufzeiten. Auf die Gesamtlaufzeit und den Median wirken sie sich nicht aus.

7.4. Verhalten in den Ausnahmefällen

In diesem Abschnitt werden die Messergebnisse und Beobachtungen zum Verhalten der drei RPA-Plattformen in definierten Ausnahmeszenarien dargelegt. Die Aufstellung dokumentiert die Reaktionen auf fachliche Geschäftsfehler, das Wiederanlaufverhalten bei transienten Systemfehlern sowie die Ergebnisse bezüglich nicht prüfbarer Fehlerklassen.

7.4.1. Geschäftsfehler: Betragsabweichung

Der Geschäftsfehler wird über den Vergleich des gebuchten mit dem ausgewiesenen Bruttobetrag erkannt. Stimmen beide innerhalb der Toleranz nicht überein, wandert die Rechnung nach Ordner 008. Alle drei Plattformen lösten diesen Mechanismus aus, jedoch unterschiedlich oft: UiPath bei zwei, Sema4.ai bei sieben und Power Automate bei sechzehn der dreißig Rechnungen. Da der Rechnungssatz identisch war, geht dieser Unterschied nicht auf die Rechnungen selbst zurück, sondern auf die Genauigkeit der jeweiligen Extraktion und Übertragung. Die zugrunde liegenden Ursachen werden in der Diskussion behandelt.

7.4.2. Systemfehler und Wiederanlauf

Das Wiederanlaufverhalten wurde in einem separaten, eigens dafür durchgeführten Durchgang geprüft. Dabei wurde die Verbindung zum ERP-System für etwa 20 Sekunden unterbrochen und anschließend wiederhergestellt. Der Test verlief bei allen drei Plattformen erfolgreich: Nach der Unterbrechung bauten sie die Verbindung zu Dolibarr erneut auf und setzten die Verarbeitung fort. Keine Rechnung ging verloren und es wurde kein Eintrag in Ordner 009 erzeugt. Die Wiederholungsintervalle waren für diesen Test bewusst kurz gehalten; im Produktivbetrieb würde man sie deutlich länger ansetzen, etwa auf zehn Minuten, um auch längere Ausfälle zu überbrücken.

Unabhängig von diesem Test trat bei Power Automate ein spontaner Abbruch auf. Bei einer Rechnung fand der Prozess ein erwartetes Feld auf der Dolibarr-Oberfläche nicht. Nach Ablauf eines Timeouts verschob er die betroffene Rechnung nach Ordner 009 und setzte mit der nächsten fort. Dieser Abbruch erklärt sowohl die Bot Success Rate von 96,7 % als auch den auffälligen Zeiteintrag dieser Rechnung. Die übrigen Rechnungen blieben davon unberührt.

7.4.3. Nicht prüfbare Fehlerklassen

Für die KI-Extraktion wurde in allen drei Implementierungen ein Wiederholungsmechanismus (Retry) auf den Aufruf des Extraktionsmodells gelegt. Ein tatsächlicher Ausfall des Extraktionsdienstes, etwa eine nicht erreichbare Erkennungs-API, ließ sich im Rahmen dieser Arbeit jedoch nicht gezielt herbeiführen: Die eingesetzten Dienste arbeiteten über alle Läufe hinweg stabil. Daher bleibt ungeprüft, ob der Retry im Fehlerfall den Dienst erneut erreicht und die Verarbeitung fortsetzt oder ob der Prozess nach erfolglosen Versuchen abbricht. Diese Fehlerklasse wird somit nicht über Messwerte abgebildet, sondern als Einschränkung ausgewiesen; die vorgesehene Behandlung ist in den Implementierungen beschrieben (Abschnitte 6.2, 6.3 und 6.4).

7.5. Implementierungsaufwand

Tabelle 11 stellt den Implementierungsaufwand der drei Prototypen gegenüber. Erfasst werden die geschätzte Entwicklungszeit, der Umfang der Lösung sowie die Art der Validierungs- und Fehlerlogik. Die Umfangszahlen sind nicht unmittelbar vergleichbar, da eine UiPath-Aktivität, eine Power-Automate-Aktion und eine Python-Codezeile unterschiedliche Granularität besitzen; sie dienen als Komplexitätsindikatoren. Die Entwicklungszeit betrug bei Sema4.ai 12 bis 14 Tage, bei UiPath 16 bis 18 Tage und bei Power Automate 25 bis 27 Tage.

Tabelle 11: Gegenüberstellung des Implementierungsaufwands.

	Sema4.ai	**UiPath**	**Power Automate**
Entwicklungszeit	12–14 Tage	16–18 Tage	25–27 Tage
Umfang	665 Codezeilen, 15 Funktionen	5 Workflows, ca. 96 Aktivitäten	15 Cloud-Aktionen; ca. 180 Desktop-Aktionen (6 Flows); 435 Zeilen PowerShell
HITL-Validierung	Dialog im Code (`RPA.Assistant`)	native Validation Station	selbstgebaute WinForms-GUI (PowerShell)
Exception-Handling	`try/except`, eigene Retry-Schleifen	Try-Catch, RetryScope, Rethrow	`retryPolicy`, `runAfter`-Zweige, If

7.6. Qualitativer Plattformvergleich

In der Methodik wurden sechs Kriterien für den qualitativen Vergleich festgelegt. Tabelle 12 hält fest, wie jede Plattform diese Kriterien im jeweiligen Prototyp umsetzt. Die Darstellung beschreibt die konkrete Realisierung; eine Bewertung der Ansätze erfolgt in der Diskussion.

Tabelle 12: Umsetzung der sechs Vergleichskriterien je Plattform.

Kriterium	UiPath	Power Automate	Sema4.ai
Geschäfts- vs. Systemfehler	Bruttobetrag-Vergleich → 008; Try-Catch → 009	If/Bruttobetrag → 008; `runAfter` (Fehler/Timeout) → 009	Bruttobetrag-Vergleich im Code → 008; `try/except` → 009
Retry-Konfigurierbarkeit	native `RetryScope` (Versuche/Intervall)	`retryPolicy` (3 Versuche, 1 min)	eigene Retry-Schleifen (5xx/Verbindung vs. 4xx)
Workflow-Handling	5 Workflows, visuell zusammengesetzt	hybrid: Cloud-Flow + 6 Desktop-Flows + PowerShell	ein Python-Programm (ca. 665 Zeilen, 15 Funktionen)
Ordner-Routing	Move-File-Aktivitäten	Verschiebe-Aktionen (Cloud/Desktop)	`pathlib-.replace()`
HITL-Validierungsstation	native Validation Station	selbstgebaute WinForms-GUI (PowerShell)	`RPA.Assistant`-Dialog im Code
Logging	Log Message + Orchestrator-Protokoll	Flow-Laufprotokoll + eigene Einträge	Logging im Code + Excel-Protokoll

7.7. Wirtschaftliche Kennzahlen

Die Wirtschaftlichkeitsmetriken werden nicht gemessen, sondern auf Basis der Formeln aus Abschnitt 5.4.1 und nachvollziehbarer Annahmen geschätzt. Tabelle 13 fasst die Annahmen samt Quellen zusammen.

Tabelle 13: Annahmen der Wirtschaftlichkeitsschätzung.

Parameter	Wert	Quelle
Manuelle Bearbeitungszeit	10 min je Rechnung	Bonpago GmbH [46]
Vollkosten-Stundensatz	ca. 42 EUR/h	Destatis 2024 [47]
Entwickler-Tagessatz	ca. 800 EUR/Tag	Freelancer-Kompass [48]
Prüfzeit je HITL-Fall	5 min	eigene Annahme
Produktive Jahresarbeitszeit	1.600 h	eigene Annahme
Wechselkurs	1 USD = 0,88 EUR (Stand 01.07.2026)	EZB-Referenzkurs [49]
HITL-Häufigkeit (Restaufwand)	10 / 50 / 100 %	gemessen (UiPath/PA/Sema4.ai)
Lizenz-/Extraktionskosten	Listenpreise	[50, 51, 52, 53]

Die Implementierungskosten ergeben sich aus Entwicklungszeit und Tagessatz: rund 10.400 EUR für Sema4.ai (13 Tage), 13.600 EUR für UiPath (17 Tage) und 20.800 EUR für Power Automate (26 Tage). Der verbleibende manuelle Aufwand nach der Automatisierung wird über die gemessene Häufigkeit der Human-in-the-Loop-Eingriffe je Plattform abgebildet – UiPath 10 %, Power Automate 50 % und Sema4.ai 100 % der Rechnungen – bei einer angenommenen Prüfzeit von fünf Minuten je Fall. Die Lizenz- und Extraktionskosten beruhen auf den öffentlichen Listenpreisen der Hersteller; die in US-Dollar angegebenen Preise wurden in Euro umgerechnet. Bei Sema4.ai entfällt eine feste Lizenz; an ihre Stelle treten nutzungsabhängige Kosten für Robocorp und die Nanonets-Extraktion, die mit dem Volumen steigen.

Tabelle 14 zeigt die Kennzahlen für drei jährliche Rechnungsvolumina. Die Volumina wurden bewusst gewählt, um ein kleines, ein mittleres und ein großes Aufkommen abzudecken und so die Skalierung der Wirtschaftlichkeit sichtbar zu machen.

Über alle drei Szenarien hinweg übersteigt der jährliche Lohnkostenvorteil die Betriebskosten. Die realen Lizenzpreise liegen durch verhandelte Mengenrabatte unter den hier verwendeten Listenpreisen; Nanonets weist solche Mengenrabatte von bis zu 40 % selbst aus [53]. Da diese Rabatte nicht öffentlich dokumentiert sind, gehen sie nicht in die Rechnung ein. Die Extraktionskosten bei UiPath (Document Understanding, über AI-Units abgerechnet) und Power Automate (in der AI-Builder-Einheit enthalten) sind in den festen

Tabelle 14: Geschätzte Wirtschaftlichkeitskennzahlen nach jährlichem Rechnungsvolumen.

Kennzahl	UiPath	Power Automate	Sema4.ai
5.000 Rechnungen pro Jahr			
Lohnkostenvorteil (EUR/Jahr)	33.250	26.250	17.500
Lizenz-/Betriebskosten (EUR/Jahr)	8.280	7.176	6.900
Netto-Nutzen (EUR/Jahr)	24.970	19.074	10.600
FTE-Einsparung	0,49	0,39	0,26
ROI (Jahr 1)	52 %	−6 %	1 %
Amortisationsdauer	6,5 Mon.	13,1 Mon.	11,8 Mon.
20.000 Rechnungen pro Jahr			
Lohnkostenvorteil (EUR/Jahr)	133.000	105.000	70.000
Lizenz-/Betriebskosten (EUR/Jahr)	8.280	7.176	27.600
Netto-Nutzen (EUR/Jahr)	124.720	97.824	42.400
FTE-Einsparung	1,98	1,56	1,04
ROI (Jahr 1)	508 %	275 %	84 %
Amortisationsdauer	1,3 Mon.	2,6 Mon.	2,9 Mon.
50.000 Rechnungen pro Jahr			
Lohnkostenvorteil (EUR/Jahr)	332.500	262.500	175.000
Lizenz-/Betriebskosten (EUR/Jahr)	8.280	7.176	69.000
Netto-Nutzen (EUR/Jahr)	324.220	255.324	106.000
FTE-Einsparung	4,95	3,91	2,60
ROI (Jahr 1)	1.420 %	838 %	120 %
Amortisationsdauer	0,5 Mon.	1,0 Mon.	1,2 Mon.

Betriebskosten nicht volumenabhängig aufgeschlüsselt; diese Einschränkung ist bei der Interpretation zu beachten.

7.8. On-Time Payment Rate (Proxy)

Die On-Time Payment Rate nach Gleichung 9 setzt die pünktlichen Zahlungen ins Verhältnis zu den Gesamtzahlungen. Sie ließ sich in dieser Arbeit nicht berechnen, da weder pünktliche Zahlungen noch Gesamtzahlungen erhoben wurden; reale Zahlungsdaten lagen nicht vor. Die in dieser Arbeit gemessenen Durchlaufzeiten lagen im Sekundenbereich je Rechnung (Median 39 bis 70 Sekunden). Den in der Literatur berichteten Zusammenhang zwischen kürzerer Durchlaufzeit und höherer Zahlungspünktlichkeit greift die Diskussion auf.

7.9. Zusammenfassung der Messergebnisse

Tabelle 15 fasst die zentralen Kennzahlen zusammen. UiPath verarbeitete den Stapel am schnellsten, mit den wenigsten manuellen Eingriffen und der höchsten tatsächlichen Autonomie. Sema4.ai legte zwar einen großen Teil der Rechnungen in Ordner 003 ab, benötigte aber bei jeder Rechnung eine Validierung. Power Automate zeigte die meisten Betragsabweichungen, den einzigen technischen Abbruch und die längste Gesamtlaufzeit.

Tabelle 15: Zusammenfassung der zentralen Messergebnisse.

Kennzahl	UiPath	Power Automate	Sema4.ai
STP-Rate (Ablage in 003)	93,3 %	43,3 %	76,7 %
Betragsabweichungen (008)	2	16	7
Technische Abbrüche (009)	0	1	0
Bot Success Rate	100,0 %	96,7 %	100,0 %
Gesamtlaufzeit	25,6 min	49,8 min	32,7 min
Median je Rechnung	39,0 s	70,1 s	56,1 s

8. Diskussion

Dieses Kapitel wertet die in Kapitel 7 dargestellten Ergebnisse aus. Es erklärt die beobachteten Unterschiede, ordnet sie in die Literatur ein und beantwortet die Forschungsfragen. Ein zentraler Gedanke zieht sich durch die Auswertung: Die drei Plattformen unterscheiden sich weniger im eigentlichen Exception-Handling-Mechanismus als in der Qualität ihrer KI-Extraktion und in der Ergonomie ihrer Umsetzung.

8.1. Extraktionsqualität als Haupttreiber der Ergebnisse

Der auffälligste Befund war die stark unterschiedliche Belegung von Ordner 008: zwei Rechnungen bei UiPath, sieben bei Sema4.ai und sechzehn bei Power Automate, obwohl alle drei denselben Rechnungssatz erhielten. Die reine Zahl legt eine dreifach schlechtere Geschäftsfehler-Erkennung bei Power Automate nahe. Bei genauer Betrachtung ist die Zahl jedoch kein Maß der Fehlererkennung, sondern ein Gemisch aus drei Ursachen. Tabelle 16 schlüsselt sie auf.

Tabelle 16: Ursachen der Ablagen in Ordner 008 (Betragsabweichung) je Plattform.

Ursache	**UiPath**	**Power Automate**	**Sema4.ai**
Beabsichtigter Geschäftsfehler	1	1	1
Modell-/Formatfehler bei der Extraktion	0	15	4
Menschlicher Fehler an der Validierung	1	0	2
Summe (Ordner 008)	2	16	7

Damit dreht sich das Bild. Die Extraktion von UiPath war fehlerfrei: Die einzige echte Abweichung war die bewusst falsch angelegte Rechnung, die zweite Ablage in 008 geht auf einen menschlichen Fehler an der Validierungsstation zurück. Bei Power Automate dagegen waren fünfzehn der sechzehn Fälle tatsächliche Extraktions- und Formatfehler. Sema4.ai liegt dazwischen: vier Modellfehler und zwei menschliche Fehler neben dem einen beabsichtigten Fall.

Die Ursache der Power-Automate-Fehler lag im AI Builder. Er interpretierte die Einzelnettobeträge wiederholt falsch, indem er Komma und Punkt vertauschte oder verschachtelte; aus einem Betrag wie „18,50“ wurde etwa „18.5000.65“, der so in das ERP-System geschrieben wurde. Erschwerend kam hinzu, dass der AI Builder gerade bei diesen groben Fehlern eine Konfidenz über der Schwelle von 80 % meldete. Die Validierungsstation wurde deshalb nicht ausgelöst, und der falsche Wert konnte nicht korrigiert werden. Ein zweiter Fehler betraf den Betragsvergleich selbst. Dolibarr stellt Beträge mit Tausenderpunkt und Komma-Cent dar, während die Testrechnungen ein gemischtes Format nutzten. Selbst bei korrekt erkannten Einzelbeträgen wich der ausgewiesene Bruttobetrag daher oft vom gebuchten ab. Eine Normalisierung wäre grundsätzlich möglich gewesen, ließ sich in der gewählten Power-Automate-Umsetzung aber nicht ohne größere Umbauten nachrüsten. Dass rund die Hälfte der Rechnungen dennoch korrekt gebucht wurde, war allein der Validierungsstation zu verdanken, in der sich die Werte im passenden Format eingeben ließen.

Bei Sema4.ai lag die Ursache nicht in falschen Werten, sondern in der Unsicherheit der Erkennung. Nanonets setzte die Feldmarkierungen mehrfach zu groß, sodass eine Positionsbeschreibung zusätzlich die Artikelnummer einschloss, ohne diese jedoch zu extrahieren. Der Bruttobetrag wurde teils gar nicht markiert, aber trotzdem korrekt ausgelesen. Diese unsauberen Markierungen dürften die niedrige Konfidenz erklären, die bei nahezu jeder Rechnung die Validierung auslöste. UiPath mit Document Understanding zeigte diese Schwächen nicht; die Extraktion war über den gesamten Korpus zuverlässig.

Zum Kontext gehört das Training. Alle drei Extraktionsmodelle wurden mit denselben 83

Rechnungen trainiert und waren zusätzlich vortrainiert. Die 30 Testrechnungen hatten die Modelle zuvor nicht gesehen. In der Entwicklungsphase kamen Rechnungen mit einheitlicher Formatierung zum Einsatz; dort lieferte auch Power Automate gute Ergebnisse, weshalb eine Normalisierung damals nicht nötig schien. Erst der heterogene Testkorpus legte die Schwäche offen. Der Befund lässt sich damit weniger als endgültiges Urteil über die Plattform lesen denn als Hinweis auf die Bedeutung der Trainings- und Testdatenvielfalt: Mit mehr und variableren Trainingsrechnungen hätte Nanonets vermutlich ein ähnlich zuverlässiges Ergebnis wie UiPath erreichen können.

8.2. STP-Rate und tatsächliche Autonomie

Die STP-Rate legt eine klare Rangfolge nahe: UiPath 93,3 %, Sema4.ai 76,7 % und Power Automate 43,3 %. Diese Rangfolge bildet die tatsächliche Autonomie jedoch nur bei UiPath ab. Die Rate zählt jede Rechnung in Ordner 003, unabhängig davon, ob ein Mensch eingreifen musste. Bei Sema4.ai löste aber jede Rechnung die Validierung aus. Die 76,7 % in Ordner 003 wurden also durchweg mit menschlicher Beteiligung erreicht; die Zahl der vollautomatisch verarbeiteten Rechnungen liegt bei null. Bei Power Automate kommt eine weitere Einschränkung hinzu: Die automatische Verschiebung korrekt gebuchter Rechnungen nach Ordner 003 schlug fehl, sodass diese Belege manuell verschoben werden mussten. Die 43,3 % waren zwar gebucht und freigegeben, der letzte Prozessschritt lief aber nicht selbstständig. Nur bei UiPath fallen Ordner-STP und echte Dunkelverarbeitung praktisch zusammen: Rund neun von zehn Rechnungen durchliefen den Prozess ohne Eingriff.

Hinter der Autonomie steht die Konfidenzschwelle, und sie wirkt in beide Richtungen. Bei Sema4.ai war die Konfidenz oft zu niedrig, obwohl die extrahierten Werte stimmten; die Folge war eine unnötige Validierung bei jeder Rechnung. Bei Power Automate war die Konfidenz umgekehrt zu hoch, obwohl die Werte falsch waren; die Folge war, dass grobe Fehler die Validierung umgingen und still ins ERP-System gelangten. Die zweite Richtung ist die gefährlichere. Eine unnötige Validierung kostet Zeit, ein nicht ausgelöster Fehler kostet Korrektheit. Eine gut kalibrierte Konfidenz ist damit für die Dunkelverarbeitung wichtiger als eine hohe Rohgenauigkeit allein.

8.3. Implementierungsaufwand: Low-Code ist nicht automatisch schneller

Der Aufwandsvergleich widerspricht der verbreiteten Annahme, Low-Code-Werkzeuge seien schneller umzusetzen als eine Programmierung im Code. Die kürzeste Entwicklungszeit erreichte mit Sema4.ai die reine Python-Lösung, die längste das Low-Code-Werkzeug Power Automate. Entscheidend war weniger die Art der Umsetzung als die Ergonomie der jeweiligen Plattform.

UiPath war am unkompliziertesten. Ordner-Routing, KI-Extraktion, Excel-Erstellung und die Fehlerbehandlung lagen als fertige Bausteine vor, teils vorbereitet, sodass sich der Prozess weitgehend zusammensetzen ließ. Das Anlegen von Variablen war für einen Informatiker intuitiv, und Document Understanding ließ sich übersichtlich einrichten und lieferte ein zuverlässiges Modell. Sema4.ai verlangte, jeden Schritt selbst zu schreiben, war dafür aber am transparentesten und dank guter Dokumentation zügig umsetzbar. Die Anbindung von Nanonets über die REST-Schnittstelle war unproblematisch, und ein Modell ließ sich schnell erstellen und trainieren; einzig die Unsicherheit der Erkennung blieb ein Schwachpunkt, dem sich mit mehr Trainingsdaten begegnen ließe.

Power Automate verursachte den größten Reibungsverlust, und zwar bereits vor der eigentlichen Programmierung. Lizenzen, Rechte, Mandanten und die unübersichtliche Rechtevergabe sowie die Verknüpfung mehrerer Dienste – Microsoft Dataverse, OneDrive und Outlook, letzteres ohne fachlichen Grund – kosteten viel Zeit. Ein lokaler Ordner ließ sich nicht nutzen, nur OneDrive. Auch die Programmierung selbst war weniger direkt als bei UiPath: Viele dort vorhandene Funktionen fehlten und mussten mühsam aus vielen

Einzelaktionen nachgebaut werden, was den großen Unterschied im Aktionsumfang erklärt. Das Zusammenspiel von Cloud- und Desktop-Flow erforderte ein ständiges Hin- und Herreichen von Variablen. Beim Training des AI Builder traten Fehler auf, nach denen sich keine Feldmarkierungen mehr setzen ließen und auch Löschen, Umbenennen und erneutes Hinzufügen nicht half; nach dem Einspielen eines Backups war das trainierte Modell zudem nicht mehr bearbeitbar. Diese Häufung erklärt die deutlich höhere Entwicklungszeit.

Beim Aufwand für das Exception-Handling im Engeren zeigt sich dasselbe Muster in kleiner Form. Bei UiPath ließ sich die Fehlerbehandlung schnell und weitgehend deklarativ über Try-Catch und RetryScope umsetzen. Bei Sema4.ai war sie vollständig auszuprogrammieren, was dank der Dokumentation aber gut gelang. Bei Power Automate war die konfigurative Umsetzung über `retryPolicy` und `runAfter` zwar machbar, aber am wenigsten intuitiv.

8.4. Wirtschaftlichkeit: Hat sich die Automatisierung gelohnt?

Die zentrale Frage der Arbeit lautet, ob sich der Automatisierungsaufwand lohnt. Die geschätzten Kennzahlen aus Tabelle 14 beantworten sie überwiegend positiv, allerdings mit einer klaren Abhängigkeit vom Rechnungsvolumen und von der Plattform. Bei mittlerem Volumen von 20.000 Rechnungen im Jahr amortisiert sich die Investition bei allen drei Lösungen innerhalb weniger Monate, und der jährliche Netto-Nutzen liegt zwischen rund 42.000 und 125.000 Euro. Bei großem Volumen verstärkt sich dieser Effekt. Nur im kleinen Szenario von 5.000 Rechnungen trägt sich Power Automate im ersten Jahr nicht; die höheren Implementierungskosten sind dort noch nicht eingespielt.

Innerhalb dieses Bildes bestätigt sich, was schon die Prozesskennzahlen andeuteten: Der entscheidende wirtschaftliche Hebel ist nicht der Lizenzpreis, sondern der verbleibende manuelle Aufwand. Er wird von der Häufigkeit der Validierungseingriffe bestimmt, und diese hängt unmittelbar an der Extraktionsqualität. UiPath schneidet am besten ab, weil kaum Eingriffe nötig waren und die Personalersparnis daher fast vollständig anfällt. Sema4.ai erzielt den geringsten Nutzen, weil die Validierung bei jeder Rechnung Arbeitszeit bindet und die nutzungsabhängigen Extraktionskosten mit dem Volumen steigen. Power Automate liegt dazwischen, wird aber durch den höchsten Implementierungsaufwand belastet.

Diese Rangfolge ist robust gegenüber der Unsicherheit bei den Lizenzpreisen. Da die Personalersparnis die Betriebskosten um ein Vielfaches übersteigt, würde selbst ein deutlich höherer oder niedrigerer Lizenzpreis das Ergebnis nicht umkehren. Reale Preise liegen durch verhandelte Mengenrabatte ohnehin unter den verwendeten Listenpreisen, was die Wirtschaftlichkeit eher verbessert. Insgesamt hat sich die Automatisierung damit gelohnt, sobald ein nennenswertes Rechnungsvolumen vorliegt – am deutlichsten dort, wo die Extraktion zuverlässig genug ist, um ohne ständige menschliche Nacharbeit auszukommen.

8.5. Imagegewinn durch schnellere Verarbeitung

Der Imagegewinn lässt sich nicht direkt messen, sondern nur über die Wirkungskette aus kürzerer Durchlaufzeit, höherer Zahlungspünktlichkeit und daraus folgender Lieferantenzufriedenheit argumentieren [vgl. 15, S. 2–4], [vgl. 16, 54]. Der erste Baustein dieser Kette ist in den Ergebnissen belegt: Eine Rechnung durchlief den automatisierten Prozess im Median in unter siebzig Sekunden, während eine manuelle Bearbeitung mehrere Minuten reine Bearbeitungszeit und im Gesamtdurchlauf oft Tage benötigt [vgl. 12, Tab. 1]. Eine solche Beschleunigung macht das Überschreiten von Zahlungsfristen und damit verbundene Mahngebühren in der Praxis unwahrscheinlich [vgl. 17, 55].

Ob daraus tatsächlich ein Reputationsvorteil erwächst, lässt sich mit den erhobenen Daten nicht belegen, da keine Zahlungsdaten vorliegen. Die Literatur stützt den Zusammenhang jedoch: Eurowag steigerte die Zahlungspünktlichkeit durch Automatisierung von 60 auf über 90 Prozent, Edenred von 6 auf 85 Prozent [17, 55]. Überträgt man diese Erfahrungswerte

auf die hier gemessenen Durchlaufzeiten, ist ein Imagegewinn plausibel, bleibt aber ein qualitativer Indikator und kein gemessenes Ergebnis. Für den Plattformvergleich ist zudem zu beachten, dass die kurze Durchlaufzeit vor allem dort entsteht, wo der Prozess ohne Validierung durchläuft – also erneut bei der Plattform mit der zuverlässigsten Extraktion.

8.6. Fallbasierte Betrachtung ausgewählter Rechnungen

Über die aggregierten Kennzahlen hinaus lohnt der Blick auf einzelne, aussagekräftige Rechnungen. Im Folgenden werden ein Standardfall und zwei Ausnahmefälle betrachtet. Jede Rechnung wird kurz vorgestellt, danach folgt das Verhalten der drei Plattformen. Diese Fälle machen anschaulich, was die Zahlen nur zusammenfassen – insbesondere, wie unterschiedlich die Plattformen mit unsicher oder falsch erkannten Attributen umgehen.

8.6.1. Standardfall: eine korrekte Rechnung

Die Rechnung des Lieferanten HolzDesign ist ein regulärer Beleg mit klar strukturierter Positionstabelle und konsistentem Bruttobetrag. Sie repräsentiert den Normalfall, für den die Automatisierung gedacht ist (siehe Abbildung 14).

UiPath verarbeitete die Rechnung ohne Eingriff korrekt und legte sie in Ordner 003 ab. Sema4.ai extrahierte die Werte zwar richtig, meldete aber eine Konfidenz unter der Schwelle und löste die Validierung aus; nach der Bestätigung landete auch diese Rechnung korrekt in 003. Power Automate ließ die Rechnung ohne Validierung durchlaufen, extrahierte sie jedoch fehlerhaft und legte sie deshalb in Ordner 008 ab. Schon der Standardfall zeigt damit das Kernmuster: nur UiPath verarbeitete vollautomatisch und korrekt, Sema4.ai benötigte den Menschen, und Power Automate scheiterte an der Extraktion.

HolzDesign Sägewerk e.K. — **OFFIZIELLE RECHNUNG**

RECHNUNGSEMPFÄNGER
MusterFirma AG
Industrieweg 5
35390 Gießen
Deutschland

Lieferantenname: HolzDesign Sägewerk e.K.
Lieferantennummer: L-30221
Rechnungsnummer: HDS-2026-991
Rechnungsdatum: 16.06.2026
Fälligkeitsdatum: 30.06.2026

Lieferung Konstruktionsholz

Sehr geehrte Damen und Herren, nachfolgend erhalten Sie die Positionsaufstellung der erbrachten Lieferungen/Leistungen:

Anzahl	Artikelnummer	Einzelnettobetrag	Positionsbeschreibung	Gesamt Netto
50 Stk.	HZ-BLH-02	14.50 €	Bauholz Fichte 40x60mmx4m	725.00 €
20 Stk.	HZ-OSB-18	22.00 €	OSB/3-Platte ungeschliffen 18mm	440.00 €

Summe Netto:	1165.00 €
USt. (19%):	221.35 €
Gesamtbruttobetrag:	**1386.35 €**

Zahlungshinweis: Bitte überweisen Sie den fälligen **Gesamtbruttobetrag** von 1386.35 € unter Angabe der Rechnungsnummer HDS-2026-991 bis zum 30.06.2026.

Bankverbindung: Sparkasse Gießen | IBAN: DE44 5135 0025 0011 2233 44 | BIC: SPKAGEDE55X
Lieferanten-Stammdaten: HolzDesign Sägewerk e.K.

Abbildung 14: Standardfall: korrekte Rechnung mit konsistentem Bruttobetrag.

8.6.2. Geschäftsfehler: falscher Bruttobetrag

Die Rechnung des Lieferanten Vogtländische Werkzeugtechnik enthält bewusst einen ausgewiesenen Bruttobetrag, der nicht zur Summe der Positionen passt (siehe Abbildung 15). Ein solcher Geschäftsfehler soll erkannt und nach Ordner 008 geroutet werden.

Alle drei Plattformen legten die Rechnung in Ordner 008 ab, das Ergebnis ist also formal in allen Fällen richtig. Der Weg dorthin unterschied sich jedoch. UiPath erkannte die Beträge korrekt und stellte die echte Abweichung fest. Sema4.ai kam nach der ausgelösten Validierung ebenfalls zum richtigen Schluss. Bei Power Automate war bereits der Einzelbetrag falsch extrahiert, sodass der Bruttovergleich zwangsläufig scheiterte – das richtige Ergebnis entstand hier aus dem falschen Grund. Der Fall verdeutlicht, dass eine korrekte Ordnerablage allein noch keine korrekte Verarbeitung belegt.

Vogtländische Werkzeugtechnik GmbH
Industriestraße 47, 08527 Plauen

RECHNUNG

Rechnungsnummer	RE-2026-44821
Rechnungsdatum	12.05.2026
Lieferantennummer	LF-3391
Fälligkeitsdatum	11.06.2026

Art.-Nr.	Beschreibung	Menge	EP netto	Summe
WZ-1042	Spannzangenfutter ER32	2 Stk	189,50 €	379,00 €
WZ-2210	HSS-Spiralbohrer-Satz 1-13 mm	3 Set	74,90 €	224,70 €
WZ-3380	Hartmetall-Wendeschneidplatten (10er)	5 Pkg	42,30 €	211,50 €
WZ-0500	Maschinenschraubstock 125 mm	1 Stk	318,00 €	318,00 €

Nettobetrag	1.133,20 €
zzgl. 19% MwSt.	215,31 €
Gesamtbetrag	**1.355,11 €**

Zahlbar innerhalb von 30 Tagen ohne Abzug.
Bankverbindung: Sparkasse Vogtland · IBAN DE56 8705 8000 0123 4567 89 · BIC WELADED1PLX · USt-IdNr. DE298745631

Abbildung 15: Geschäftsfehler: ausgewiesener Bruttobetrag weicht von der Positionssumme ab.

8.6.3. Fehlextraktion im Anzahl-Feld: „21 km á 0,45 €“

Die Rechnung des Lieferanten Elektro Meyer enthält in der Position „Fahrtkosten“ im Anzahl-Feld nicht eine reine Zahl, sondern den Text „21 km á 0,45 €“. Dieses Feld ist bewusst so gestaltet, dass es die Extraktion in die Irre führt und einen nicht-numerischen Wert erzeugt (siehe Abbildung 16). Der Fall ist der aufschlussreichste der Arbeit, weil er direkt zeigt, wie gut sich ein falsch extrahiertes Attribut noch abfangen lässt.

Elektro Meyer & Sohn GmbH
Werkstattweg 8, 34123 Kassel

RECHNUNG

Rechnungsnummer	RE-2026-50914
Rechnungsdatum	19.05.2026
Lieferantennummer	LF-2274
Leistungsdatum	19.05.2026
Fälligkeitsdatum	02.06.2026

Beschreibung	Anzahl	EP netto	Summe
Arbeitszeit Monteur	4 Std	58,00 €	232,00 €
Fahrtkosten / Anfahrt	20 km à 0,45 €	—	9,00 €
Kleinmaterial pauschal	1 Pos	24,50 €	24,50 €
Prüfung Elektroinstallation (DGUV V3)	1 Pos	45,00 €	45,00 €

Nettobetrag	310,50 €
zzgl. 19% MwSt.	58,99 €
Gesamtbetrag	**369,49 €**

Zahlbar innerhalb von 14 Tagen ohne Abzug.
Bankverbindung: Kasseler Sparkasse · IBAN DE34 5205 0353 0000 4455 66 · BIC HELADEF1KAS · USt-IdNr. DE813344556

Abbildung 16: Fehlerprovokation: nicht-numerischer Wert im Anzahl-Feld.

Alle drei Plattformen extrahierten dieses Feld falsch. Erst danach trennten sich die Wege, und zwar entlang der beiden Bedingungen, unter denen eine Validierung greift: ob sie ausgelöst wird und ob der Mensch den Fehler bemerkt. Bei UiPath lag die Konfidenz unter der Schwelle, die Validierung wurde ausgelöst, und der falsche Wert konnte korrigiert werden Somit landete die Rechnung korrekt in Ordner 003. Bei Sema4.ai wurde die Validierung ebenfalls ausgelöst, doch der falsche Wert wurde beim Prüfen übersehen und irrtümlich bestätigt; die Rechnung wurde mit dem falschen Wert gebucht. Bei Power Automate lag die Konfidenz trotz des groben Fehlers über der Schwelle, die Validierung wurde gar nicht ausgelöst, und der falsche Wert gelangte unbemerkt in das ERP-System.

Derselbe Fehler führte also zu drei verschiedenen Ausgängen: abgefangen und korrigiert, abgefangen aber menschlich übersehen, gar nicht abgefangen. Bemerkenswert ist, dass dieselbe Rechnung bei UiPath und Sema4.ai eine niedrige, bei Power Automate hingegen eine hohe Konfidenz erhielt; der AI Builder war bei einem eindeutig falschen Wert überzuversichtlich. Der Fall zeigt damit die Grenzen der Human-in-the-Loop-Validierung: Sie schützt nur, wenn sie erstens ausgelöst wird und zweitens der Mensch aufmerksam ist. Fällt die erste Bedingung weg, wird der Fehler nie sichtbar; fällt die zweite weg, hilft auch die beste Konfidenzschätzung nicht.

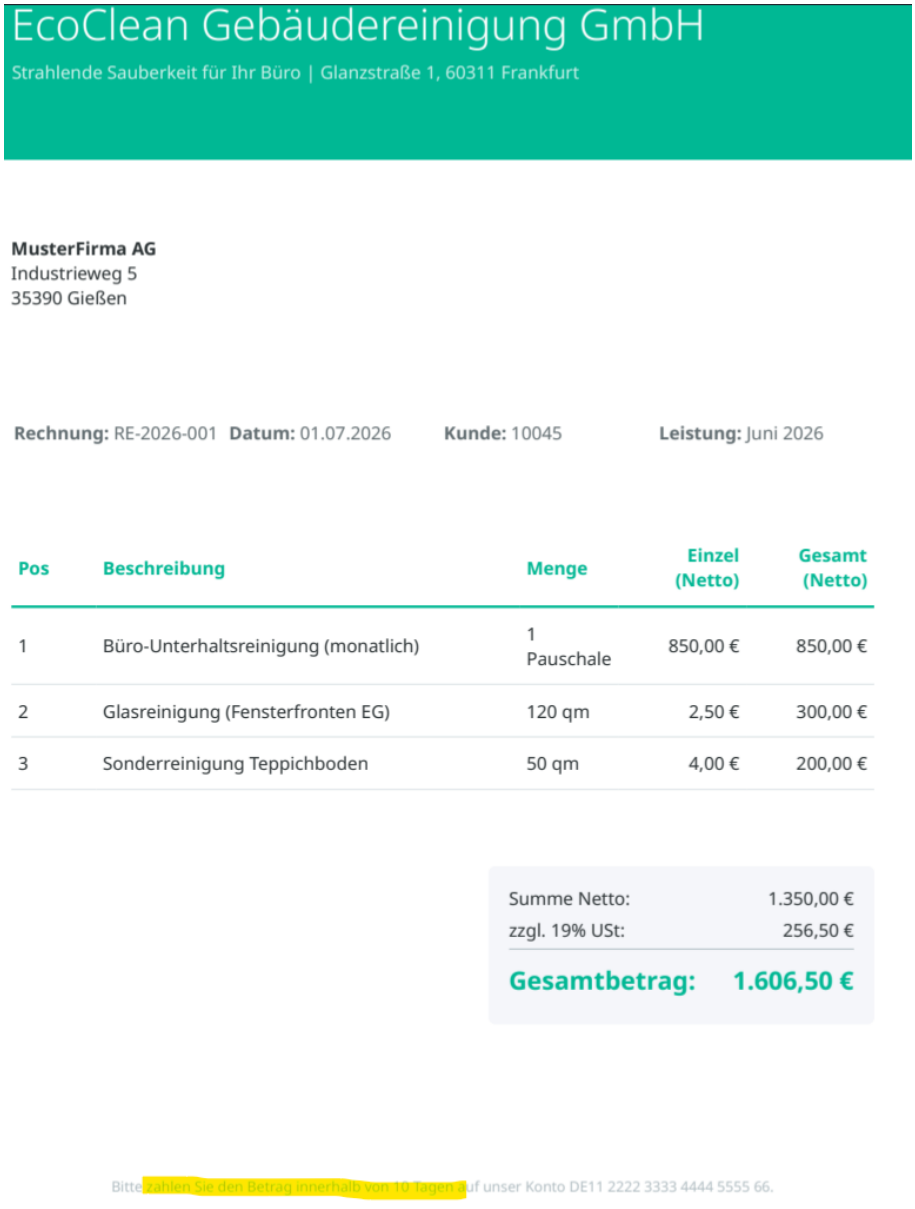
8.6.4. Fälligkeitsdatum in Textform: „In 10 Tagen bitte zahlen“

Die letzte betrachtete Rechnung nennt das Fälligkeitsdatum nicht als Datum, sondern als Fließtext: Statt eines Werts wie „24.06.2026“ steht dort „In 10 Tagen bitte zahlen“. Ein solches Feld lässt sich nicht unmittelbar als Datum auslesen und stellt die Extraktion vor eine andere Herausforderung als ein grob falscher Zahlenwert (siehe Abbildung 17).

Im Test verhielten sich alle drei Plattformen gleich. Die Extraktion des Fälligkeitsdatums war auf jeder Plattform unsicher, sodass die Konfidenz unter die Schwelle fiel und die Validierung ausgelöst wurde. Nach der manuellen Eingabe des korrekten Datums ließen sich alle drei Rechnungen erfolgreich buchen; sie landeten in Ordner 003.

Dieser Fall ist das Gegenstück zum vorigen. Beim Anzahl-Feld glitt bei Power Automate

ein falscher Wert mit hoher Konfidenz durch; hier führte die Textform des Datums auf allen drei Plattformen zu einer niedrigen Konfidenz, sodass die Validierung zuverlässig auslöste und der Fehler auf jeder Plattform korrigierbar war. Der Fall zeigt damit die positive Seite des Konfidenzmechanismus: Erkennt das Modell seine eigene Unsicherheit, hält die Validierungsstation den Prozess an, lässt die Korrektur zu und führt die Rechnung in den automatisierten Ablauf zurück. Ein ungewöhnliches, aber für einen Menschen lesbares Feld ist damit weniger gefährlich als ein falscher Wert, der mit falscher Sicherheit geliefert wird.



EcoClean Gebäudereinigung GmbH

Strahlende Sauberkeit für Ihr Büro | Glanzstraße 1, 60311 Frankfurt

MusterFirma AG
Industrieweg 5
35390 Gießen

Rechnung: RE-2026-001 **Datum:** 01.07.2026 **Kunde:** 10045 **Leistung:** Juni 2026

Pos	Beschreibung	Menge	Einzel (Netto)	Gesamt (Netto)
1	Büro-Unterhaltsreinigung (monatlich)	1 Pauschale	850,00 €	850,00 €
2	Glasreinigung (Fensterfronten EG)	120 qm	2,50 €	300,00 €
3	Sonderreinigung Teppichboden	50 qm	4,00 €	200,00 €

Summe Netto: 1.350,00 €
zzgl. 19% USt: 256,50 €

Gesamtbetrag: 1.606,50 €

Bitte zahlen Sie den Betrag innerhalb von 10 Tagen auf unser Konto DE11 2222 3333 4444 5555 66.

Abbildung 17: Fälligkeitsdatum in Textform statt als Datum.

8.7. Einordnung in die Literatur

Die gemessenen Werte lassen sich an den in der Methodik genannten Branchen-Benchmarks spiegeln. Die durchschnittliche STP-Rate liegt laut Ardent Partners bei rund 68,9 % [12]. UiPath übertrifft diesen Wert mit 93,3 % deutlich und nähert sich dem Spitzenbereich, während Power Automate mit 43,3 % klar darunter bleibt. Die 76,7 % von Sema4.ai liegen zwar über dem Durchschnitt, entsprechen aber, wie gezeigt, keiner echten Dunkelverarbeitung. Die durchschnittliche Exception Rate von etwa 18,4 % [12] wird von UiPath unterschritten

und von den beiden anderen Plattformen überschritten. Die Bot Success Rate von 100 % bei UiPath und Sema4.ai sowie 96,7 % bei Power Automate liegt nahe an den in der Praxis berichteten Werten von über 98 % [13]. Die geschätzten Wirtschaftlichkeitskennzahlen bewegen sich bei mittlerem und großem Volumen im Bereich der berichteten ROI-Werte von durchschnittlich rund 250 % und darüber [56].

Im Verhältnis zu den verwandten Arbeiten bestätigt sich die Einordnung aus Kapitel 4. Die vorliegende Arbeit verbindet den empirischen Plattformvergleich mit einem konkreten, KI-gestützten und ERP-integrierten Rechnungsprozess und legt den Schwerpunkt auf die Ausnahmebehandlung. Diese Kombination fehlt in den betrachteten Arbeiten, die entweder eine einzelne Plattform umsetzen [35] oder Plattformen nur anhand von Merkmalslisten vergleichen [39]. Zugleich bleibt die vorliegende Arbeit in der Datenmenge hinter groß angelegten Studien zurück, etwa der produktiv eingesetzten Lösung von Tater et al. mit rund 80.000 Rechnungen [38].

8.8. Herausforderungen bei der Umsetzung

Neben den auswertbaren Kennzahlen traten während der Umsetzung mehrere Probleme auf, die für die Bewertung der Plattformen aufschlussreich sind und die hier gesondert festgehalten werden.

Bei Power Automate schlug die automatische Verschiebung korrekt gebuchter Rechnungen nach Ordner 003 fehl, obwohl dieselbe Aktion für die Ablage nach Ordner 008 funktionierte und nur der Zielpfad angepasst war. Die Freigabe im ERP-System gelang, allein der abschließende Verschiebeschritt nicht. Die betroffenen dreizehn Rechnungen mussten daher von Hand verschoben werden. Abbildung 18 zeigt den betroffenen Ausschnitt des Flows.

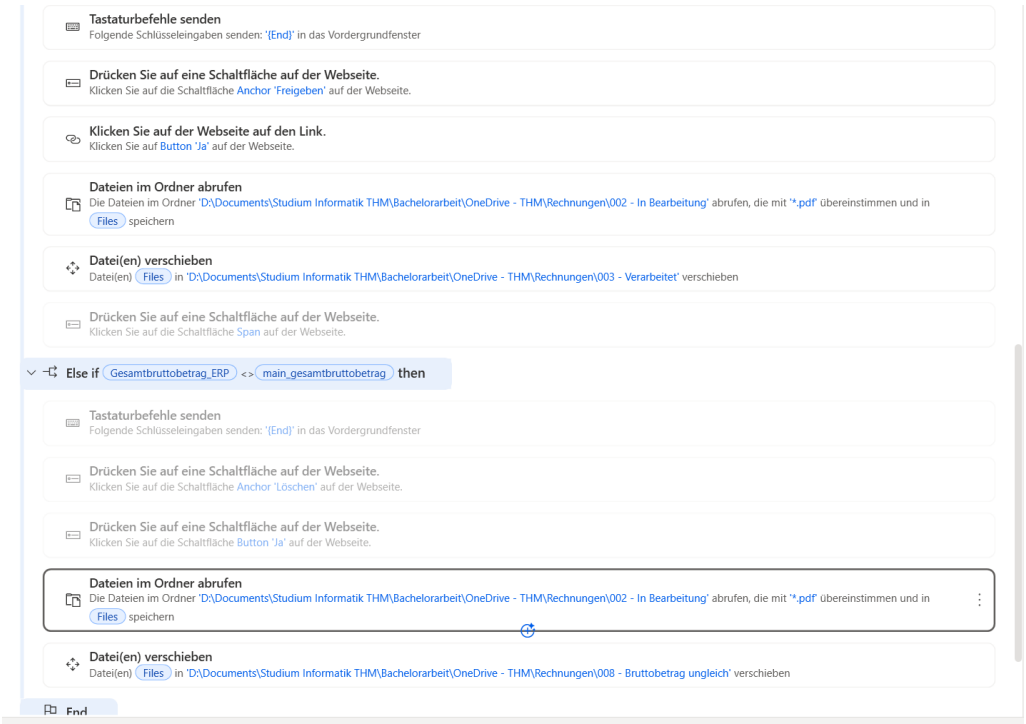


Abbildung 18: Power Automate: fehlgeschlagener Verschiebeschritt nach Ordner 003 im If-Zweig.

Die zweite Herausforderung betraf die Extraktion des AI Builder. Er interpretierte die Dezimaltrennung der Einzelnettobeträge falsch und meldete dafür zugleich eine hohe Konfidenz. Genau diese Kombination ist die Hauptursache der vielen Ablagen in Ordner 008, denn die falschen Werte umgingen die Validierung. Erschwerend kam hinzu, dass Power

Automate keine einsatzbereite Validierungsstation bereitstellte, sodass diese in PowerShell selbst gebaut werden musste. Anders als die Validierungsstation von UiPath oder die Review-Oberfläche von Nanonets markierte diese Eigenentwicklung die erkannten Felder nicht sichtbar auf der Rechnung; fehlerhafte Werte ließen sich beim manuellen Prüfen dadurch deutlich schwerer erkennen. Abbildung 19 zeigt beispielhaft die Markierung und die fehlerhafte Interpretation der Einzelnettobeträge im AI Builder.

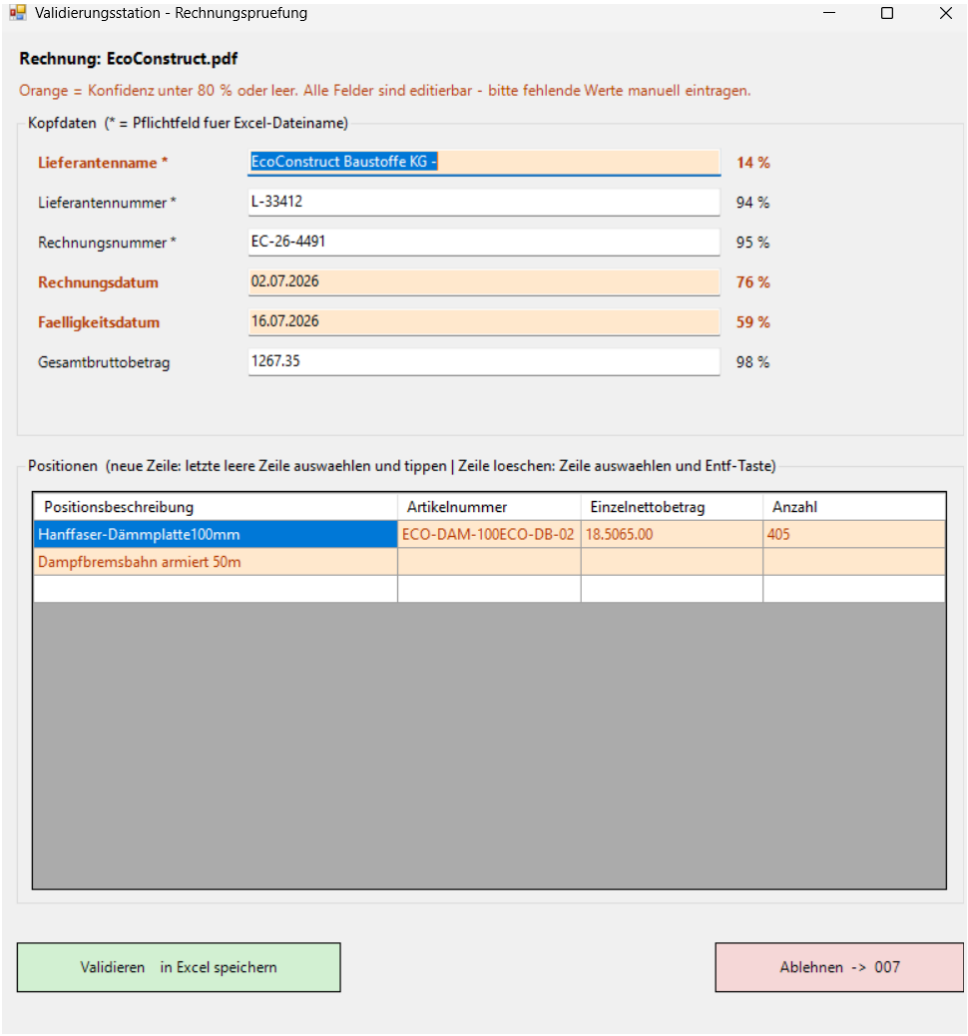


Abbildung 19: Power Automate / AI Builder: fehlerhafte Interpretation der Dezimaltrennung.

Die dritte Herausforderung zeigte sich bei Sema4.ai. Nanonets setzte die Feldmarkierungen mehrfach zu groß und schloss benachbarte Felder mit ein, extrahierte den Wert aber dennoch korrekt; den Bruttobetrag markierte es teils gar nicht, las ihn aber richtig aus. Abbildung 20 zeigt eine solche Fehlmarkierung, die die niedrige Konfidenz und damit die häufige Validierung erklärt.

Abbildung 20: Sema4.ai / Nanonets: fehlerhafte Feldmarkierung bei dennoch korrekter Extraktion.

Hinzu kamen bei Power Automate wiederkehrende Probleme abseits des Prozesses selbst: Nach Trainingsfehlern ließen sich keine Feldmarkierungen mehr setzen, und nach dem Einspielen eines Backups war das Modell nicht mehr bearbeitbar. Solche Werkzeugprobleme fließen in keine Kennzahl ein, prägen aber den praktischen Aufwand erheblich.

8.9. Beantwortung der Forschungsfragen

Auf Grundlage der Ergebnisse und ihrer Deutung lassen sich die Forschungsfragen beantworten.

Die **Teilfrage 1.1** nach den erforderlichen Implementierungsschritten beantwortet der in Kapitel 6 beschriebene Prozess: einmalige Anmeldung am ERP-System, schleifenweise Bearbeitung der Rechnungen mit KI-Extraktion, Konfidenzprüfung mit gegebenenfalls ausgelöster Human-in-the-Loop-Validierung, Buchung in Dolibarr, Abgleich des Bruttobetrags und Ablage im entsprechenden Zielordner. Diese Schritte sind auf allen drei Plattformen gleich; nur ihre technische Realisierung unterscheidet sich.

Die **Teilfrage 1.2** nach den auftretenden Ausnahmen lässt sich klar aufteilen. Geschäftsfehler traten als Betragsabweichungen (Ordner 008) auf; ein Sonderfall wäre die Ablehnung in der Validierung (Ordner 007), die im Test nicht vorkam. Systemfehler traten als technischer Abbruch (Ordner 009) auf, einmal spontan bei Power Automate; ein provozierter ERP-Ausfall wurde vom Wiederanlaufmechanismus aufgefangen. Die Fallbetrachtung ergänzt eine dritte, praktisch bedeutsame Kategorie: die still bleibende Fehlbuchung, bei der

ein falscher Wert mit hoher Konfidenz die Ausnahmebehandlung umgeht.

Die **Teilfrage 1.3** nach dem Aufwand für das Exception-Handling beantwortet der Aufwandsvergleich: am geringsten und weitgehend deklarativ bei UiPath, mittel und selbst ausprogrammiert, aber gut dokumentiert bei Sema4.ai, am aufwendigsten und am wenigsten intuitiv bei Power Automate.

Die **Teilfrage 1.4** nach der Wirtschaftlichkeit beantworten die geschätzten Kennzahlen: Die Automatisierung amortisiert sich bei mittlerem und großem Volumen bei allen drei Plattformen innerhalb weniger Monate, mit dem höchsten Netto-Nutzen bei UiPath. Der entscheidende Hebel ist der von der Extraktionsqualität abhängige Validierungsaufwand, nicht der Lizenzpreis.

Die **Teilfrage 1.5** nach der Validierbarkeit und Korrigierbarkeit eines unsicher oder falsch extrahierten Attributs beantwortet vor allem die Fallbetrachtung. Ein unsicheres, aber korrektes Attribut löst die Validierung aus und lässt sich problemlos bestätigen – bei Sema4.ai geschah das unnötig oft. Ein falsches Attribut ist nur dann korrigierbar, wenn die Validierung ausgelöst wird und der Mensch den Fehler bemerkt. Beide Bedingungen können scheitern: Bei Power Automate blieb die Validierung wegen zu hoher Konfidenz aus, bei Sema4.ai wurde der Fehler menschlich übersehen. Die Rückführung in den automatisierten Prozess gelang nur dort zuverlässig, wo die Konfidenz gut kalibriert war und der Bearbeiter aufmerksam blieb.

Die **Hauptforschungsfrage** nach den Unterschieden der Plattformen im Exception-Handling hinsichtlich der Wirtschaftlichkeit lässt sich damit zusammenfassen: Die Plattformen unterscheiden sich im eigentlichen Fehlerbehandlungsmechanismus weniger als in der Qualität ihrer KI-Extraktion und der Ergonomie ihrer Umsetzung. Diese beiden Faktoren bestimmen, wie oft ein Mensch eingreifen muss, und der manuelle Aufwand wiederum bestimmt die Wirtschaftlichkeit. UiPath war wirtschaftlich am günstigsten, weil die zuverlässige Extraktion kaum Eingriffe verlangte; Sema4.ai war transparent und schnell umzusetzen, litt aber unter unsicherer Extraktion; Power Automate war am aufwendigsten in Einrichtung und Umsetzung und in der Extraktion am fehleranfälligsten.

8.10. Limitationen

Die Ergebnisse sind vor dem Hintergrund mehrerer Einschränkungen zu lesen. Der Rechnungskorpus ist konstruiert und mit dreißig Rechnungen klein; die Anteile sind daher illustrativ und nicht statistisch abgesichert, zumal je Plattform nur ein Lauf erfolgte. Alle Tests liefen auf einem einzelnen Rechner, alle Daten wurden in das gleiche ERP-System eingetragen und die Validierung nahm eine einzige Person vor, wodurch menschliche Fehler wie im Anzahl-Fall unmittelbar in die Ergebnisse eingehen. Die Wirtschaftlichkeitswerte sind Schätzungen auf Basis offengelegter Annahmen; insbesondere die Lizenzpreise beruhen auf Listenpreisen, während reale Preise durch nicht öffentlich dokumentierte Mengenrabatte darunter liegen. Ein echter Ausfall des Extraktionsdienstes ließ sich nicht provozieren, sodass offen bleibt, ob der dort hinterlegte Retry den Dienst wieder erreicht oder der Prozess abbricht. Die Durchlaufzeit enthält Validierungswartezeiten und taugt nur als grober Indikator. Schließlich beziehen sich die Extraktionsbefunde auf den Stand der eingesetzten Dienste zum Testzeitpunkt; mit mehr Trainingsdaten oder neueren Modellversionen können sie anders ausfallen. Eine weitere Einschränkung betrifft den Wiederanlauf bei nicht-atomaren Operationen. Läuft das ERP-System beim zweiten Versuch wieder, obwohl beim ersten Versuch bereits ein Teil des Belegs angelegt wurde, kann eine Doppelung entstehen. Für eine produktionsreife Lösung müsste vor jedem Wiederholungsversuch geprüft werden, ob die Rechnung bereits existiert; das würde die Fehlerbehandlung deutlich verkomplizieren und wurde im Rahmen dieser Arbeit nicht umgesetzt.

9. Fazit

Ziel dieser Arbeit war die vergleichende Analyse des zeitlichen und technischen Implementierungsaufwands von Exception-Handling-Mechanismen auf ausgewählten RPA-Plattformen. Da die Automatisierung in der Praxis selten am fehlerfreien Regelfall, sondern an der komplexen Behandlung von Ausnahmen scheitert, bestimmt die Modellierung dieser Fehlerpfade maßgeblich die Wirtschaftlichkeit solcher IT-Projekte. Um dies empirisch zu untersuchen, wurde ein identischer, KI-gestützter Rechnungsverarbeitungsprozess auf drei technologisch unterschiedlichen Plattformen – UiPath (Low-Code), Microsoft Power Automate (Cloud-Low-Code) und Sema4.ai (Pro-Code/Python) – implementiert und in das quelloffene ERP-System Dolibarr integriert. Ein gemeinsamer Testkorpus von 30 Rechnungen durchlief jede Lösung unter identischen Bedingungen, wobei eine einheitliche Konfidenzschwelle von 80 % für die Human-in-the-Loop-Validierung und eine statusbasierte Ordnerablage als Messinstrumente dienten.

Die zentrale Erkenntnis der Untersuchung lautet, dass sich die Plattformen im eigentlichen informationstechnischen Exception-Handling-Mechanismus (wie etwa Try-Catch-Blöcken oder automatisierten Retry-Schleifen) weitaus weniger unterscheiden als in der Qualität ihrer KI-Extraktion und der Ergonomie ihrer Umsetzung. Beide Faktoren bestimmen primär, wie oft ein menschlicher Sachbearbeiter korrigierend eingreifen muss, was wiederum den verbleibenden manuellen Aufwand und damit die tatsächliche Wirtschaftlichkeit der Automatisierungslösung diktiert.

Hinsichtlich der Extraktionsqualität und Prozesseffizienz lieferte UiPath (mit Document Understanding) die besten Resultate: Die Plattform verarbeitete den Korpus mit einer Straight-Through-Processing-Rate (STP-Rate) von 93,3 % nahezu vollautomatisch, wies die geringste Fehlerquote auf und war mit einer medianen Durchlaufzeit von 39 Sekunden je Rechnung am schnellsten. Power Automate (mit dem AI Builder) erzielte hingegen mit 43,3 % die niedrigste STP-Rate und produzierte die meisten Geschäftsfehler, da Dezimaltrennzeichen teils falsch interpretiert wurden. Erschwerend kam hinzu, dass das Modell für diese Fehlinterpretationen eine hohe Konfidenz (über 80 %) ausgab, wodurch die fehlerhaften Werte die manuelle Validierung umgingen und stillschweigend ins ERP-System gebucht wurden. Sema4.ai (mit Nanonets) erreichte auf dem Papier zwar eine erfolgreiche Ablagequote von 76,7 %, erzwang jedoch aufgrund großzügiger Feldmarkierungen und durchgängig niedriger Konfidenzwerte bei 100 % der Rechnungen eine manuelle Validierung. Die tatsächliche Dunkelverarbeitung lag hier folglich bei null.

Der gemessene Implementierungsaufwand widerlegte zudem das verbreitete Postulat, dass visuelle Low-Code-Plattformen zwingend eine schnellere Entwicklung ermöglichen als klassische Programmieransätze. Die reine Code-Lösung Sema4.ai (Python) ließ sich dank klarer Strukturierung und transparenter Dokumentation mit 12 bis 14 Tagen Entwicklungszeit am schnellsten umsetzen. UiPath erwies sich mit 16 bis 18 Tagen als hochgradig intuitiv, da native Bausteine wie die Validation Station direkt integriert waren. Power Automate verursachte mit 25 bis 27 Tagen Entwicklungszeit die größten Reibungsverluste. Komplexe Rechtevergaben, die hybride Architektur aus Cloud- und Desktop-Flows sowie das Fehlen einer nativen, einsatzbereiten Validierungsoberfläche erzwangen die Entwicklung eines eigenen Workarounds über ein 435 Zeilen langes PowerShell-Skript, was den Aufwand massiv in die Höhe trieb.

Bezüglich der Wirtschaftlichkeit zeigte die modellierte Schätzung, dass sich die Automatisierung ab einem mittleren Volumen (z. B. 20.000 Rechnungen pro Jahr) bei allen drei Plattformen innerhalb von wenigen Monaten amortisiert. Der entscheidende wirtschaftliche Hebel ist dabei nicht der anfängliche Software-Lizenzpreis, sondern der von der Extraktionsqualität abhängige personelle Validierungsaufwand.

Abschließend schließt diese Arbeit eine relevante Forschungslücke: Sie verbindet den Plattformvergleich mit einem ERP-integrierten End-to-End-Prozess und beweist, dass ein stabiles Exception Handling für eine gewinnbringende Prozessautomatisierung bilden.

10. Ausblick

Aus den Grenzen und Beobachtungen dieser Arbeit ergeben sich mehrere Anknüpfungspunkte für weiterführende Untersuchungen. Ein naheliegender Schritt betrifft den Rechnungseingang selbst. Statt die Belege manuell in den Eingangsordner zu legen, ließen sie sich direkt aus einem E-Mail-Postfach beziehen, etwa über Outlook oder einen anderen Mail-Dienst. Die eingehenden Rechnungen könnten dann automatisch im Eingangsordner gesammelt oder unmittelbar verbucht werden, wodurch der Prozess durchgängiger würde.

Eine zweite Richtung kehrt den Prozess um. Statt eingehende Lieferantenrechnungen zu verarbeiten, ließe sich die Automatisierung auf die Erstellung eigener Ausgangsrechnungen übertragen. Damit würde ein verwandter, aber gegenläufiger Geschäftsprozess untersucht, der andere Anforderungen an Datenquellen und Validierung stellt.

Darüber hinaus wären die quantitativen Ergebnisse durch einen größeren und realen Rechnungskorpus, durch mehr Testläufe und durch mehrere Bearbeiter statistisch besser abzusichern; der Einbezug weiterer Plattformen würde den Vergleich verbreitern. Da sich die Extraktionsqualität als entscheidender Faktor erwiesen hat, wäre insbesondere eine umfangreichere und variablere Trainingsbasis für Nanonets vielversprechend. Offen bleibt außerdem das Verhalten bei einem echten Ausfall des Extraktionsdienstes: Ob der hinterlegte Wiederholungsmechanismus den Dienst wieder erreicht oder der Prozess abbricht, ließe sich in einem gezielten Fehlertest klären. Schließlich würde die On-Time Payment Rate erst mit realen Zahlungsdaten von einem literaturgestützten Proxy zu einer belastbaren Messgröße.

Abbildungsverzeichnis

1. Dolibarr ERP System. [18] . . . 10
2. UiPath RPA. [20] . . . 11
3. Microsoft Power Automate. [25] . . . 12
4. Sema4.ai. [30] . . . 13
5. Überblick über das Anwendungsszenario der KI-gestützten Rechnungsverarbeitung. . . . 18
6. Abstrahierter Prozessaufbau der KI-gestützten Rechnungsverarbeitung mit integrierten Verzweigungen zur Fehlerbehandlung und statusbasierter Ergebnisablage. . . . 22
7. Standardprozess in UiPath: Grundgerüst (oben) und Extraktionsschritte mit äußerer Fehlerbehandlung (unten). . . . 30
8. Verschachtelte Try-Catch-Struktur des UiPath-Prototyps (aus `Main.xaml`). 31
9. Manuelle Validierung in UiPath über die integrierte Validation Station. . . . 32
10. Ablauf des Power-Automate-Prototyps mit Zuordnung der Schritte zu den drei Schichten (Cloud, Desktop, PowerShell). . . . 34
11. Selbstgebaute Validierungsstation in Power Automate (PowerShell/WinForms). 36
12. Logging in Power Automate über den automatischen Cloud-Laufverlauf. . . . 37
13. Architektur des Sema4.ai-Prototyps und Anbindung der externen Schnittstellen (Nanonets, Dolibarr, Excel). . . . 38
14. Standardfall: korrekte Rechnung mit konsistentem Bruttobetrag. . . . 49
15. Geschäftsfehler: ausgewiesener Bruttobetrag weicht von der Positionssumme ab. . . . 50
16. Fehlerprovokation: nicht-numerischer Wert im Anzahl-Feld. . . . 51
17. Fälligkeitsdatum in Textform statt als Datum. . . . 52
18. Power Automate: fehlgeschlagener Verschiebeschritt nach Ordner 003 im If-Zweig. . . . 53
19. Power Automate / AI Builder: fehlerhafte Interpretation der Dezimaltrennung. 54
20. Sema4.ai / Nanonets: fehlerhafte Feldmarkierung bei dennoch korrekter Extraktion. . . . 55
21. UiPath Main-Workflow Teil 1. . . . 66
22. UiPath Main-Workflow Teil 2. . . . 67
23. UiPath Main-Workflow Teil 3. . . . 68
24. UiPath Main-Workflow Teil 4. . . . 69
25. UiPath Main-Workflow Teil 5. . . . 70
26. UiPath Main-Workflow Teil 6. . . . 71

Tabellenverzeichnis

1. Einordnung der verwandten Arbeiten (+ vorhanden, o ansatzweise, – nicht behandelt). . . . 15
2. Im Anwendungsszenario ausgelesene Rechnungsfelder. . . . 17
3. Statusverzeichnisse zur Ablaufsteuerung und Ergebnisablage. . . . 17
4. Gegenüberstellung der ausgewählten RPA-Plattformen. . . . 18
5. Übersicht der ausgewählten Metriken und ihre Zuordnung zu den Forschungsfragen. . . . 26
6. Technische Komponenten und Konfiguration der Testumgebung. . . . 26
7. Systematisierung und Erhebungswege der Evaluationsmetriken. . . . 27
8. Verteilung der 30 Rechnungen auf die Endordner je Plattform. . . . 41
9. Prozesseffizienz-Metriken je Plattform (Anteile in Prozent). . . . 41
10. Durchlaufzeiten je Plattform (erfasst mit `dlz_watcher.py`). . . . 42
11. Gegenüberstellung des Implementierungsaufwands. . . . 43
12. Umsetzung der sechs Vergleichskriterien je Plattform. . . . 44
13. Annahmen der Wirtschaftlichkeitsschätzung. . . . 44
14. Geschätzte Wirtschaftlichkeitskennzahlen nach jährlichem Rechnungsvolumen. 45
15. Zusammenfassung der zentralen Messergebnisse. . . . 45
16. Ursachen der Ablagen in Ordner 008 (Betragsabweichung) je Plattform. . . . 46

Listings

1. Power Automate – Retry über `retryPolicy` und Fehler-Routing über `runAfter` (Auszug aus der Flow-Definition). 35
2. Power Automate – Kern der selbstgebauten Validierungsstation (PowerShell/WinForms): Schwellwertprüfung und die zwei Schaltflächen. 36
3. Power Automate – Verschiebung nach 003, nach erfolgreichen Durchlauf des Power Automate Desktop Flows. 37
4. Power Automate – Verschiebung nach 009, nach gescheiterten Durchlauf des Power Automate Desktop Flows. 37
5. Sema4.ai – Hauptablauf der Pipeline (gekürzt): Statuswechsel 001 → 002, KI-Extraktion, optionale Validierung, Buchung und Ablage in 003/008/009. 38
6. Sema4.ai – Retry mit Unterscheidung der Fehlerklassen: 5xx und Verbindungsfehler werden wiederholt, 4xx bricht sofort ab. 39
7. Sema4.ai – Konfidenzprüfung: sobald ein Feld oder eine Zelle unter 80 % liegt, wird die Validierung ausgelöst. 39
8. Sema4.ai – Human-in-the-Loop-Dialog über `RPA.Assistant`; "Abbrechen" verschiebt nach 007, "Weiter" ruft die korrigierten Werte erneut ab. 40
9. Sema4.ai – Logging über `print` und Ergebnis-Routing über `pathlib.replace`. 40
10. Python-Skript `dlz_watcher.py` zur laufzeitgenauen Erfassung der Durchlaufzeit. 72
11. Python-Skript `auswertung_ordner.py` zur Berechnung der Prozesseffizienz-Metriken aus der Ordnerstruktur. 74

Literatur

1. Syed R, Suriadi S, Adams M, Bandara W, Leemans SJJ, Ouyang C, Hofstede AHM ter, Weerd I van de, Wynn MT und Reijers HA. Robotic Process Automation: Contemporary themes and challenges. Computers in Industry 2020; 115. DOI: 10.1016/j.compind.2019.103162
2. Gronau N. Enterprise Resource Planning: Architektur, Funktionen und Management von ERP-Systemen. 4. Berlin: Gito, 2021
3. Weber P, Gabriel R, Lux T und Menke K. Basiswissen Wirtschaftsinformatik. 4. Aufl. Wiesbaden: Springer Vieweg, 2022. DOI: 10.1007/978-3-658-35616-3
4. Yashwant S, Dubey A, Paikray P und Thulsiram G. Invoice Information Extraction: Methods and Performance Evaluation. arXiv preprint 2025. ManpowerGroup Services India Pvt. Ltd.; Indian Institute of Technology, Hyderabad. Zugriff am 23.06.2026. arXiv: 2510.15727 [cs.AI]. Available from: https://arxiv.org/abs/2510.15727
5. Ho Vo Hoang D, Vo Quoc H und Hung BT. ConBGAT: A Novel Model Combining Convolutional Neural Networks, Transformer and Graph Attention Network for Information Extraction from Scanned Image. PeerJ Computer Science 2024; 10. DOI: 10.7717/peerj-cs.2536
6. Wewerka J und Reichert M. Robotic Process Automation – A Systematic Literature Review and Assessment Framework. arXiv preprint 2021. Institute of Databases and Information Systems, Ulm University, Zugriff am 19.06.2026. arXiv: 2012.11951 [cs.SE]. Available from: https://arxiv.org/abs/2012.11951
7. Abdellaif OH, Nader A und Hamdi A. ERPA: Efficient RPA Model Integrating OCR and LLMs for Intelligent Document Processing. *2024 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*. IEEE, 2024. DOI: 10.48550/arXiv.2412.19840
8. UiPath. Business Exception vs. Application Exception. Zugriff am 03.06.2026. 2023. Available from: https://docs.uipath.com/orchestrator/standalone/2023.4/user-guide/business-exception-vs-application-exception
9. Hohpe G und Woolf B. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. The Addison-Wesley Signature Series. Addison-Wesley Professional, 2003
10. Deutschland. Gesetz zur Stärkung von Wachstumschancen, Investitionen und Innovation sowie Steuervereinfachung und Steuerfairness (Wachstumschancengesetz). Bundesgesetzblatt Teil I, Nr. 108. 2024
11. Huda M, Rahayu A, Furqon C, Sultan MA und Sugiana NSS. Analyzing the Impact of Robotic Process Automation (RPA) on Productivity and Firm Performance in the Service Sector. International Journal of Advanced Computer Science and Applications 2025; 16. DOI: 10.14569/IJACSA.2025.0160635
12. Ardent Partners. State of ePayables 2025. Branchenbericht. Ardent Partners, 2025. Available from: https://www.sap.com/documents/2026/01/72ee9140-3e7f-0010-bca6-c68f7e60039b.html
13. Quille RVE, Almeida FVd, Borycz J, Corrêa PLP, Filgueiras LVL, Machicao J, Almeida GMd, Midorikawa ET, Demuner VRdS, Bedoya JAR u. a. Performance Analysis Method for Robotic Process Automation. Sustainability 2023; 15. DOI: 10.3390/su15043702
14. Gadatsch A. Grundkurs Geschäftsprozessmanagement: Methoden und Werkzeuge für die IT-Praxis. 7. Wiesbaden: Springer Vieweg, 2012

15. Gavrila SG, Blanco González-Tejero C, Gómez Gandía JA und Lucas Ancillo A de. The Impact of Automation and Optimization on Customer Experience: A Consumer Perspective. Humanities and Social Sciences Communications 2023; 10. DOI: 10.1057/s41599-023-02389-0

16. Isaac Construction. Isaac Construction Builds Reputation. Unternehmens-Fallstudie. 2022. Available from: https://istart.co.nz/nz-case-studies/isaac-construction-fujifilm-accounts-payable-automation/

17. Rossum. Eurowag Achieves 70 % Invoice Automation Across 60 Entities with Rossum and Coupa Integration. Unternehmens-Fallstudie. 2026. Available from: https://rossum.ai/case-studies/

18. Dolibarr. Dolibarr Open Source ERP and CRM. Zugriff am 22.05.2026. 2026. Available from: https://www.dolibarr.org/

19. Li G, González López de Murillas E, Carvalho RM de und Aalst WMP van der. Extracting Object-Centric Event Logs to Support Process Mining on Databases. *Business Information Systems (BIS 2017)*. Springer, 2017. DOI: 10.1007/978-3-319-59336-4_4

20. UiPath. LandingPage. Zugriff am 25.06.2026. 2026. Available from: https://www.uipath.com

21. Gartner. Magic Quadrant for Robotic Process Automation. 2025 Jun. Available from: https://ir.uipath.com/news/detail/400/uipath-recognized-as-a-leader-in-the-2025-gartner-magic-quadrant-for-robotic-process-automation

22. Microsoft. Handling and throwing exceptions in .NET. 2026. Available from: https://learn.microsoft.com/en-us/dotnet/standard/exceptions/

23. UiPath. Studio — Robotic Enterprise Framework. UiPath Documentation, Studio Standalone 2024.10. Zugriff am 22.06.2026. 2024. Available from: https://docs.uipath.com/studio/standalone/2024.10/user-guide/robotic-enterprise-framework

24. UiPath. Document Understanding – Validation Station. Zugriff am 25.05.2026. 2026. Available from: https://docs.uipath.com/document-understanding/automation-cloud/latest/user-guide/validation-station

25. baitechdata. was ist powerAutomate. Zugriff am 25.06.2026. 2026. Available from: https://baitechdata.de/was-ist-power-automate/

26. Microsoft. Employ robust error handling - Power Automate. Microsoft Learn. Zugriff am 15.06.2026. 2025. Available from: https://learn.microsoft.com/en-us/power-automate/guidance/coding-guidelines/error-handling

27. Microsoft. Overview of adaptive cards for Teams - Power Automate. Microsoft Learn. Zugriff am 15.06.2026. 2023. Available from: https://learn.microsoft.com/en-us/power-automate/overview-adaptive-cards

28. Microsoft. about_Error_Handling - PowerShell. Microsoft Learn. Zugriff am 15.06.2026. 2026. Available from: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_error_handling?view=powershell-7.6

29. Microsoft. Invoice processing prebuilt AI model. Microsoft Learn. Zugriff am 15.06.2026. 2025. Available from: https://learn.microsoft.com/en-ie/ai-builder/prebuilt-invoice-processing

30. Sema4.ai. LandingPage. Sema4.ai Website. Zugriff am 26.06.2026. 2026. Available from: https://sema4.ai/

31. Sema4.ai. Documentation. Sema4.ai Documentation. Zugriff am 16.06.2026. 2026. Available from: https://docs.sema4.ai/

32. Python Software Foundation. Errors and Exceptions — Python 3.12 Documentation. 2026. Available from: https://docs.python.org/3/tutorial/errors.html

33. Robocorp. Robocorp documentation. Robocorp documentation. Zugriff am 16.06.2026. 2026. Available from: https://robocorp.com/docs/

34. Nanonets. Invoice OCR API & Invoice OCR Software. Zugriff am 22.05.2026. 2026. Available from: https://nanonets.com/ocr-api/invoice-ocr

35. Santos M und Trigo A. Automation of Invoice Processing with ERP Integration Using RPA Tools. *Atas da 23ª Conferência da Associação Portuguesa de Sistemas de Informação (CAPSI)*. 2023. DOI: 10.18803/capsi.v23.36-52

36. Desai D, Jain A, Naik D, Panchal N und Sawant D. Invoice Processing using RPA & AI. SSRN Electronic Journal 2021. DOI: 10.2139/ssrn.3852575

37. Rohaime NA, Abdul Razak NI, Thamrin NM und Shyan CW. Integrated Invoicing Solution: A Robotic Process Automation with AI and OCR Approach. *IEEE Student Conference on Research and Development (SCOReD)*. 2022. DOI: 10.1109/SCOReD57082.2022.9973841

38. Tater T, Gantayat N, Dechu S u. a. AI Driven Accounts Payable Transformation. *Proceedings of the AAAI Conference on Artificial Intelligence*. Bd. 36. 2022. DOI: 10.1609/aaai.v36i11.21506

39. Sindhuja R, Modugu PT, Goud SA, Kumar ER, Babu GS und Reddy R. A Comparative Analysis of RPA Tools: UiPath, Automation Anywhere and Robocorp. *2024 OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0*. 2024. DOI: 10.1109/OTCON60325.2024.10688237

40. Khan S. Comparative Analysis of RPA Tools – UiPath, Automation Anywhere and BluePrism. International Journal of Computer Science and Mobile Applications 2020; 8. DOI: 10.47760/ijcsma.2020.v08i11.001

41. Bornegrim L und Holmquist G. Robotic process automation - An evaluative model for comparing RPA-tools. Bachelor Thesis, Uppsala University. Zugriff am 22.06.2026. 2020. Available from: https://www.diva-portal.org/smash/get/diva2:1442991/FULLTEXT01.pdf

42. Microsoft. Vorgefertigtes Modell zur Rechnungsverarbeitung in AI Builder verwenden. Zugriff am 22.05.2026. 2026. Available from: https://learn.microsoft.com/de-de/ai-builder/prebuilt-invoice-processing

43. Sema4.ai. From RPA to Python Automation (Robocorp). Zugriff am 22.05.2026. 2026. Available from: https://sema4.ai/docs/automation/faq/to-python-automation

44. Aalst WMP van der, Bichler M und Heinzl A. Robotic Process Automation. Business & Information Systems Engineering 2018; 60. DOI: 10.1007/s12599-018-0542-4

45. UiPath. Exception Handling – UiPath Studio Documentation. Zugriff am 10.06.2026. 2025. Available from: https://docs.uipath.com/studio/standalone/2023.10/user-guide/exception-handling

46. Bonpago GmbH. Unterschied XRechnung und ZUGFeRD. Zugriff am 25.06.2026. 2023. Available from: https://www.bonpago.de/blog/tag/e-rechnung/unterschied-xrechnung-und-zugferd

47. Statistisches Bundesamt (Destatis). Eine Arbeitsstunde kostete im Jahr 2024 durchschnittlich 43,40 Euro. Pressemitteilung Nr. 154 vom 30.04.2025. Marktbestimmte Dienstleistungen: 42,10 EUR je geleistete Stunde. 2025. Available from: https://www.destatis.de/DE/Presse/Pressemitteilungen/2025/04/PD25_154_624.html.Zugriff%20am%2020.06.2026

48. freelancermap GmbH. Freelancer-Kompass 2025. Zugriff am 20.06.2026. 2025. Available from: https://www.freelancermap.de/marktstudie

49. Europäische Zentralbank. Euro-Referenzkurs: US-Dollar (USD). 1 EUR = 1,1406 USD. Zugriff am 01.07.2026. 2026. Available from: https://www.ecb.europa.eu/stats/policy_and_exchange_rates/euro_reference_exchange_rates/html/eurofxref-graph-usd.de.html

50. UiPath. Plans and Pricing. Zugriff am 20.06.2026. 2025. Available from: https://www.uipath.com/pricing

51. Microsoft. Power Automate Pricing. Zugriff am 20.06.2026. 2025. Available from: https://www.microsoft.com/en/power-platform/products/power-automate/pricing

52. Sema4.ai. RPA Pricing. Zugriff am 20.06.2026. 2025. Available from: https://sema4.ai/docs/automation/pricing

53. Nanonets. Pricing. Zugriff am 20.06.2026. 2025. Available from: https://nanonets.com/pricing

54. Procurement Leaders. Southern Water – A Case Study in Sourcing Delivery. Unternehmens-Fallstudie. 2018. Available from: https://barkersprocurement.com/case-study-southern-water/

55. Precoro Inc. How Edenred went from 6% to 85% on-time payments with Precoro. Precoro Website. Customer Story / Fallstudie. Zugriff am 18.06.2026. 2026. Available from: https://precoro.com/customers/customer-cases/edenred-case-study

56. Parker H und Appel SE. On the Path to Artificial Intelligence: The Effects of a Robotics Solution in a Financial Services Firm. South African Journal of Industrial Engineering 2021; 32. DOI: 10.7166/32-2-2390

A. Anhang

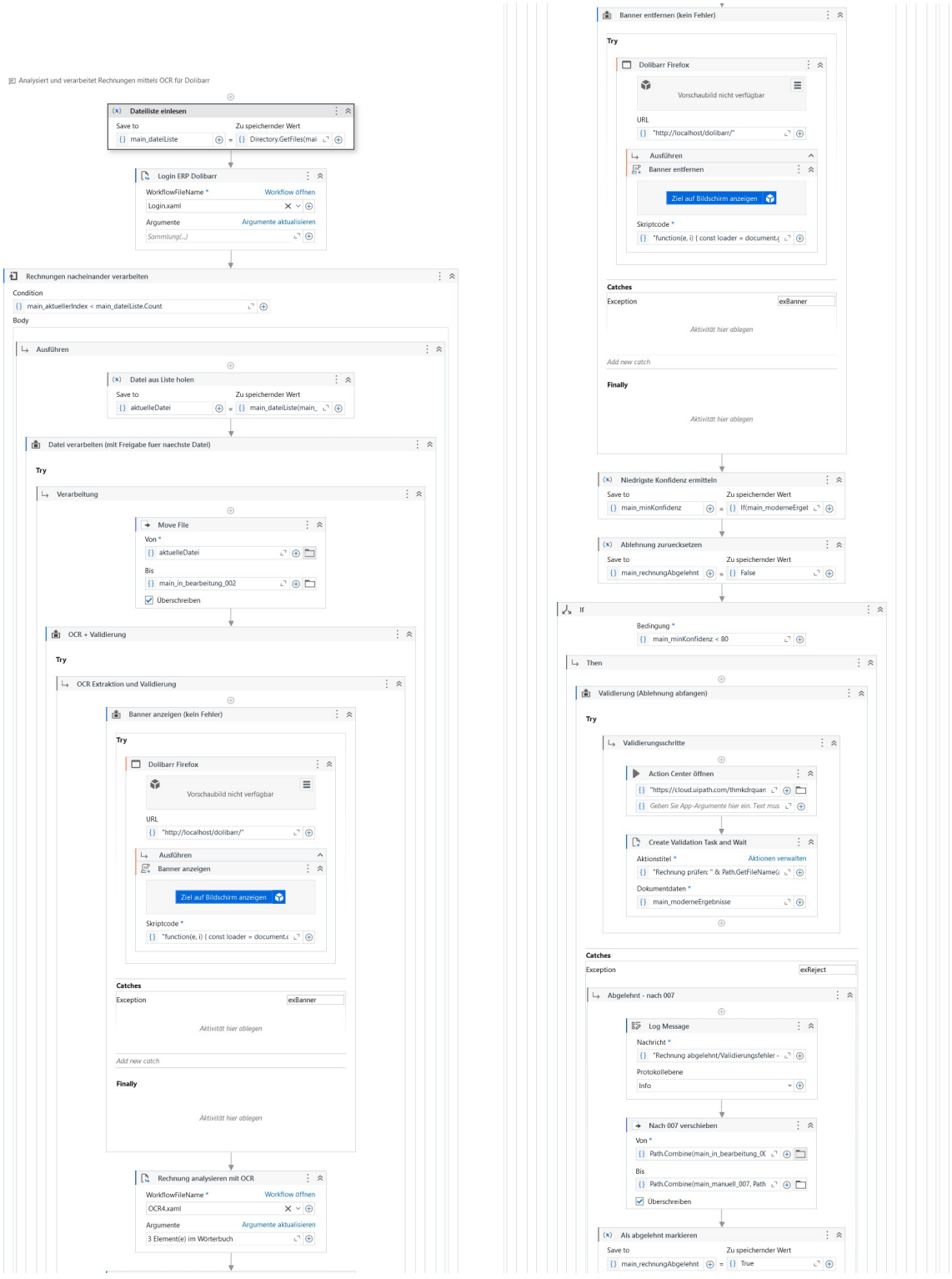


Abbildung 21: UiPath Main-Workflow Teil 1.

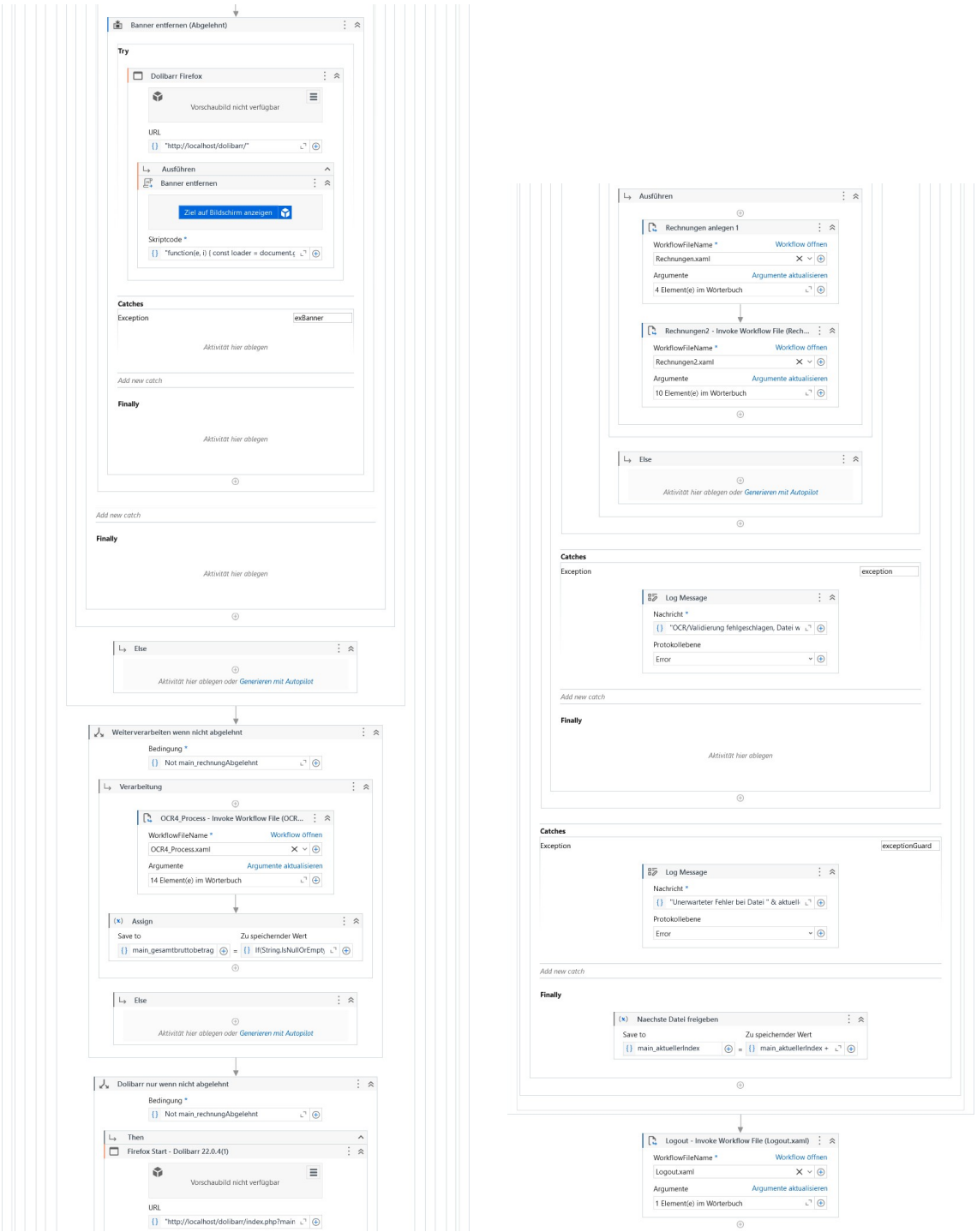


Abbildung 22: UiPath Main-Workflow Teil 2.

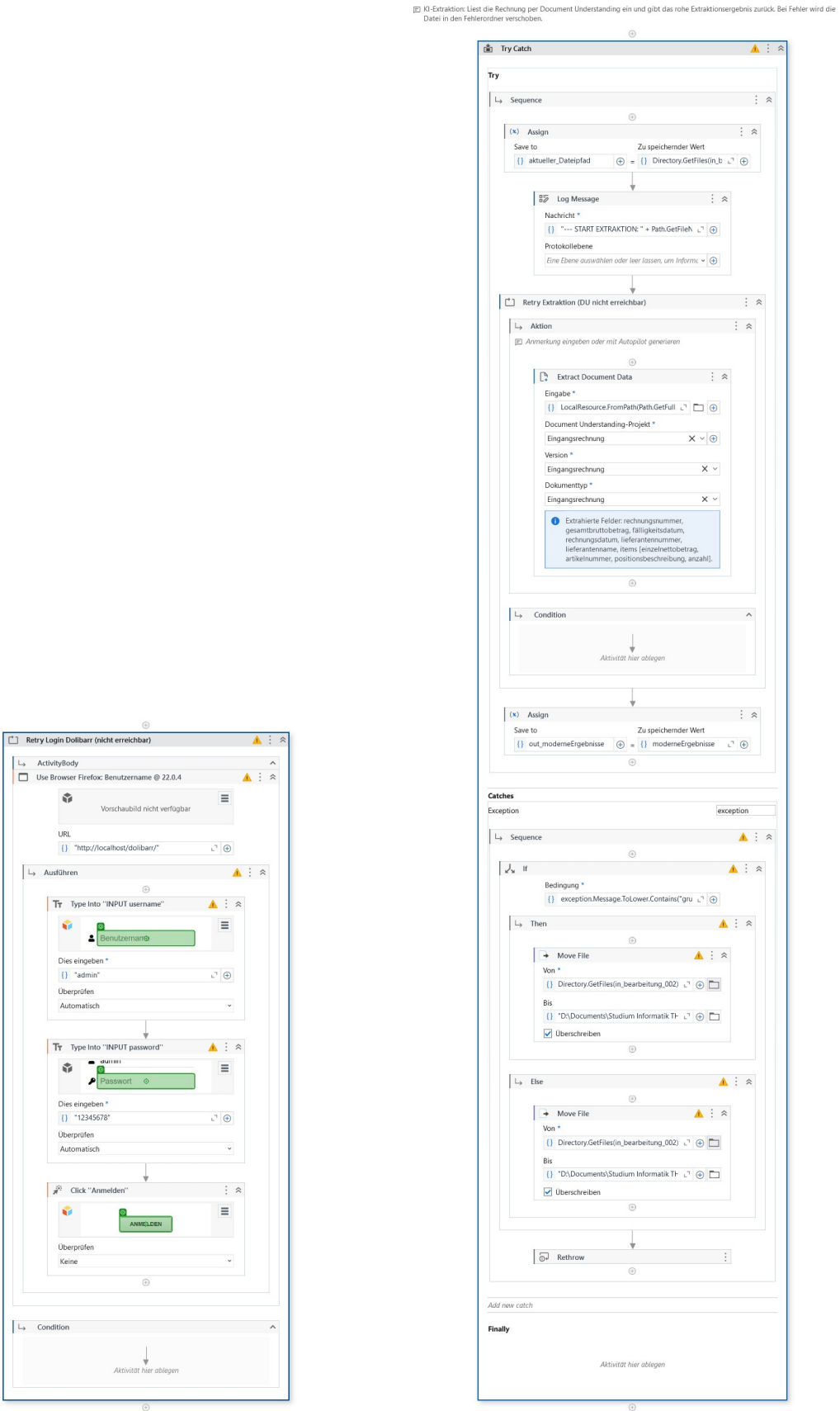
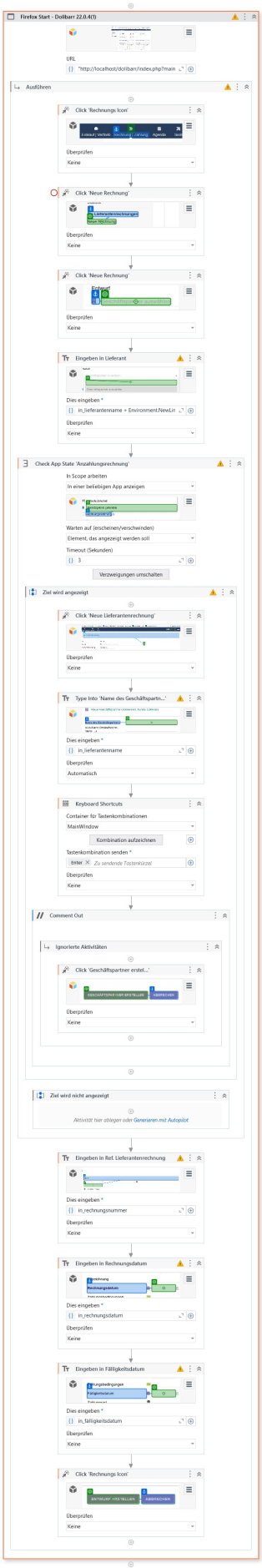


Abbildung 23: UiPath Main-Workflow Teil 3.

Abbildung 24: UiPath Main-Workflow Teil 4.



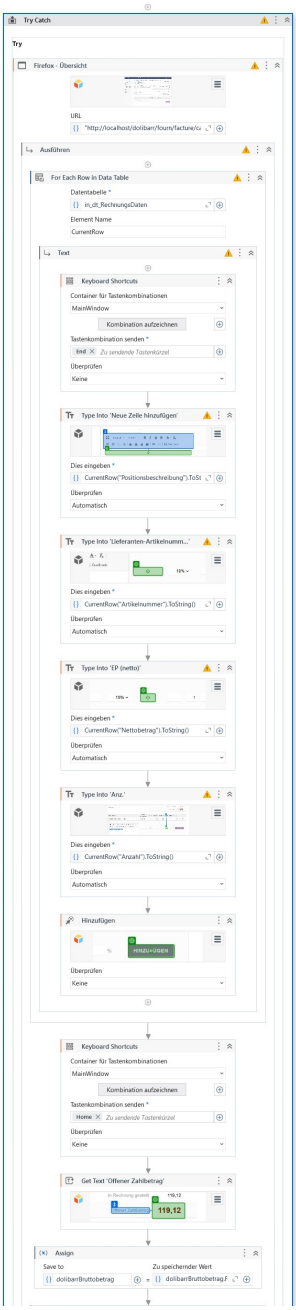
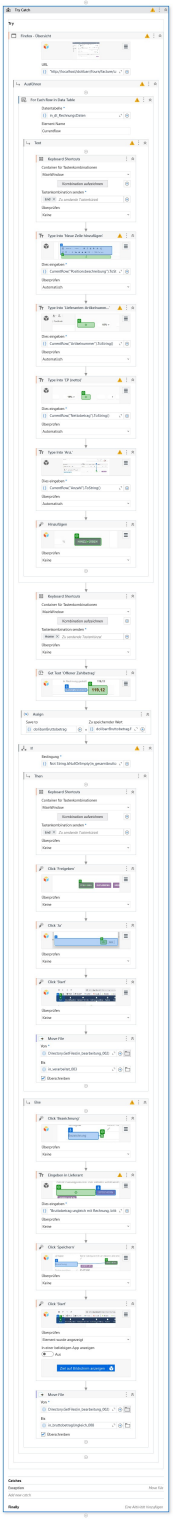


Abbildung 25: UiPath Main-Workflow Teil 5.



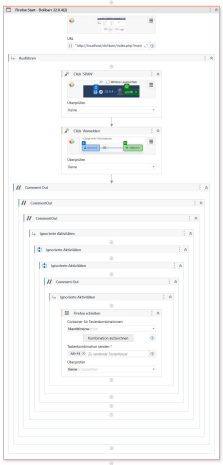


Abbildung 26: UiPath Main-Workflow Teil 6.

B. Anhang

```python
import sys
import csv
import time
import signal
import statistics
from pathlib import Path
from datetime import datetime

# === KONFIGURATION ====================================================
BASIS = Path(r"D:\Documents\Studium Informatik THM\Bachelorarbeit\Rechnungen")
PLATTFORM = "UiPath"

POLL = 2.0            # Sekunden zwischen zwei Pruefungen
STOP_GRACE = 30       # Sek.: so lange muessen 001 & 002 leer sein, dann Stopp

ENDORDNER = [
    "003 - Verarbeitet",
    "007 - Rechnungen manuell überprüfen",
    "008 - Bruttobetrag ungleich",
    "009 - Fehler",
]
OFFEN = ["001 - Unbearbeitet", "002 - In Bearbeitung"]
# ======================================================================

CSV_NAME = None
for i, a in enumerate(sys.argv):
    if a == "--csv" and i + 1 < len(sys.argv):
        CSV_NAME = sys.argv[i + 1]


def pdfs_in(name):
    ordner = BASIS / name
    if not ordner.is_dir():
        return set()
    return {p.name for p in ordner.iterdir()
            if p.is_file() and p.suffix.lower() == ".pdf"}


def offen_leer():
    return all(len(pdfs_in(o)) == 0 for o in OFFEN)


seen = {}          # dateiname -> timestamp
start = time.time()
gestoppt = {"flag": False}


def beenden(*_):
    gestoppt["flag"] = True


signal.signal(signal.SIGINT, beenden)

# Bestand vor dem Lauf merken (wird nicht mitgezaehlt)
vorbestand = set()
for o in ENDORDNER:
    vorbestand |= pdfs_in(o)

print(f"\n=== Durchlaufzeit-Mitschreiber: {PLATTFORM} ===")
print(f"Basis: {BASIS}")
print(f"Start: {datetime.fromtimestamp(start).strftime('%Y-%m-%d %H:%M:%S')}")
if vorbestand:
```

```
    print(f"[Hinweis] {len(vorbestand)} Datei(en) lagen schon in Endordnern "
          f"und werden ignoriert.")
print("Jetzt den Bot starten. Beenden: Strg + C oder automatisch am Ende.\n")
print(f"{'Uhrzeit':9} {'+ s seit Start':>14}  Datei")
print("-" * 64)

leer_seit = None
while not gestoppt["flag"]:
    jetzt = time.time()
    for o in ENDORDNER:
        for n in pdfs_in(o):
            if n in vorbestand or n in seen:
                continue
            seen[n] = jetzt
            print(f"{datetime.fromtimestamp(jetzt).strftime('%H:%M:%S'):9} "
                  f"{jetzt - start:14.1f}  {n}")
    # Auto-Stopp, wenn 001 & 002 laenger leer sind
    if seen and offen_leer():
        leer_seit = leer_seit or jetzt
        if jetzt - leer_seit >= STOP_GRACE:
            print(f"\n[Auto-Stopp] 001 und 002 seit {STOP_GRACE}s leer.")
            break
    else:
        leer_seit = None
    time.sleep(POLL)

# ---- Auswertung ----
if not seen:
    print("\nKeine neuen Rechnungen erfasst.")
    sys.exit(0)

eintraege = sorted(seen.items(), key=lambda kv: kv[1])  # nach Zeit
zeilen = []
prev = start
for name, ts in eintraege:
    zeilen.append([name, ts, ts - start, ts - prev])
    prev = ts

einzel = [z[3] for z in zeilen]
median = statistics.median(einzel)
gesamt = eintraege[-1][1] - start

print("\n=== Kennzahlen ===")
print(f"Rechnungen ................... {len(zeilen)}")
print(f"Gesamtlaufzeit (Stapel) ...... {gesamt/60:6.1f} min")
print(f"Median Bearbeitungszeit ...... {median:6.1f} s")
print(f"Durchschnitt ................. {statistics.mean(einzel):6.1f} s")
print(f"Minimum / Maximum ............ {min(einzel):.1f} s / {max(einzel):.1f} s")

ziel_csv = Path(CSV_NAME) if CSV_NAME else BASIS / f"dlz_{PLATTFORM.replace(' ', '_')}.csv
    "
with open(ziel_csv, "w", newline="", encoding="utf-8") as f:
    w = csv.writer(f, delimiter=";")
    w.writerow(["Datei", "Fertig_um", "Sek_seit_Start", "Einzelzeit_s"])
    for name, ts, seit, einzelz in zeilen:
        w.writerow([name, datetime.fromtimestamp(ts).strftime("%H:%M:%S"),
                    f"{seit:.1f}", f"{einzelz:.1f}"])
print(f"\nCSV geschrieben: {ziel_csv}")
```

Listing 10: Python-Skript `dlz_watcher.py` zur laufzeitgenauen Erfassung der Durchlaufzeit.

```python
import sys
import csv
from pathlib import Path

# === KONFIGURATION ============================================================
# Basis-Ordner anpassen. Beispiel Sema4.ai:
BASIS = Path(r"D:\Documents\Studium Informatik THM\Bachelorarbeit\Rechnungen")

# Plattform-Name (nur fuer die Ausgabe / CSV)
PLATTFORM = "UiPath"

# Ordnernamen exakt wie im Dateisystem.
ORDNER = {
    "001": "001 - Unbearbeitet",
    "002": "002 - In Bearbeitung",
    "003": "003 - Verarbeitet",
    "007": "007 - Rechnungen manuell überprüfen",
    "008": "008 - Bruttobetrag ungleich",
    "009": "009 - Fehler",
}
# ==============================================================================


def zaehle_pdfs(ordner: Path) -> int:
    """Zaehlt PDF-Dateien (Gross-/Kleinschreibung egal) in einem Ordner."""
    if not ordner.is_dir():
        return 0
    return sum(1 for p in ordner.iterdir()
               if p.is_file() and p.suffix.lower() == ".pdf")


def prozent(zaehler: int, nenner: int) -> float:
    return (zaehler / nenner * 100) if nenner else 0.0


def main():
    zahl = {key: zaehle_pdfs(BASIS / name) for key, name in ORDNER.items()}

    # Endzustaende = abgeschlossene Rechnungen
    gesamt = zahl["003"] + zahl["007"] + zahl["008"] + zahl["009"]

    print(f"\n=== Auswertung: {PLATTFORM} ===")
    print(f"Basis: {BASIS}\n")
    print(f"{'Ordner':32} {'PDFs':>6}")
    print("-" * 40)
    for key, name in ORDNER.items():
        print(f"{name:32} {zahl[key]:6d}")
    print("-" * 40)
    print(f"{'Abgeschlossen (003+007+008+009)':32} {gesamt:6d}")

    # Hinweis, falls noch Rechnungen offen sind
    if zahl["001"] or zahl["002"]:
        print(f"\n[Hinweis] Noch offen: {zahl['001']} in 001, {zahl['002']} in 002 "
              f"(noch nicht verarbeitet).")

    if gesamt == 0:
        print("\n[Abbruch] Keine abgeschlossenen Rechnungen gefunden. "
              "Bitte BASIS-Pfad pruefen.")
        return

    # --- Metriken ---
    stp        = prozent(zahl["003"], gesamt)
    exception  = prozent(zahl["007"] + zahl["008"] + zahl["009"], gesamt)
    bot_success = prozent(gesamt - zahl["009"], gesamt)
```

```
65
66      business_exc = zahl["007"] + zahl["008"]   # fachliche Ausnahmen
67      system_exc   = zahl["009"]                 # technische Ausnahmen
68
69      print("\n=== Metriken ===")
70      print(f"STP-Rate ............. {stp:6.1f} %   ({zahl['003']}/{gesamt})")
71      print(f"Exception Rate ...... {exception:6.1f} %   ({zahl['007']+zahl['008']+zahl
            ['009']}/{gesamt})")
72      print(f"  davon Business ........ {prozent(business_exc, gesamt):6.1f} %   "
73            f"(007: {zahl['007']}, 008: {zahl['008']})")
74      print(f"  davon System .......... {prozent(system_exc, gesamt):6.1f} %   "
75            f"(009: {zahl['009']})")
76      print(f"Bot Success Rate .... {bot_success:6.1f} %   ({gesamt - zahl['009']}/{gesamt})
            ")
77      print()
78
79      # --- optionaler CSV-Export ---
80      if "--csv" in sys.argv:
81          ziel = Path(sys.argv[sys.argv.index("--csv") + 1])
82          with open(ziel, "w", newline="", encoding="utf-8") as f:
83              w = csv.writer(f, delimiter=";")
84              w.writerow(["Plattform", "003", "007", "008", "009", "Gesamt",
85                          "STP_%", "Exception_%", "Business_%", "System_%", "BotSuccess_%"])
86              w.writerow([PLATTFORM, zahl["003"], zahl["007"], zahl["008"], zahl["009"],
87                          gesamt, f"{stp:.1f}", f"{exception:.1f}",
88                          f"{prozent(business_exc, gesamt):.1f}",
89                          f"{prozent(system_exc, gesamt):.1f}", f"{bot_success:.1f}"])
90          print(f"CSV geschrieben: {ziel}\n")
91
92
93  if __name__ == "__main__":
94      main()
```

Listing 11: Python-Skript `auswertung_ordner.py` zur Berechnung der Prozesseffizienz-Metriken aus der Ordnerstruktur.