



Konzeption einer Datenspeicherlösung zur Erstellung von RPA-Robotern in verschiedenen Modellierungsoberflächen

Design of a data storage solution for the creation of RPA robots in different modeling interfaces

Lukas Hüller

Prof. Dr. Mathias Weske
Maximilian Völker, MSc.

Fachgebiet für Business Process Technology

Datum der Abgabe: 28.06.2021

Ich erkläre hiermit, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Potsdam, 27. Juni 2021

Lukas Hüller

Abstract. Die Robotic Process Automation (RPA) hat in den vergangenen Jahren im Bereich der Automatisierungslösungen zunehmend an Bedeutung gewonnen. Der Markt stellt inzwischen eine Vielzahl von Plattformen zur Verfügung, mit denen die Automatisierungen erstellt werden können. Fast alle dieser Plattformen besitzen proprietäre Oberflächen zur Erstellung der Roboter, wodurch ein ständiges Übersetzen der modellierten und dokumentierten Prozesse aus bekannten Modellierungssprachen wie die Business Process Model and Notation (BPMN) oder Flowcharts in die Darstellungsform der Anwendungen notwendig ist. Dadurch erhöht sich der Aufwand für RPA-Entwickler und Prozessexperten in der Erstellung der Roboter.

Die vorliegende Arbeit untersucht die für die Erstellung von RPA-Robotern geeigneten Notationen. Auf dieser Grundlage wird eine Architektur vorgestellt, die das Speichern der Roboter in einer einheitlichen Datenspeicherlösung ermöglicht. Zudem wird gezeigt, wie die verschiedenen Repräsentationen aus dieser „*Single Source of Truth*“ (SSoT) generiert werden können. Mit der vorgestellten Lösung ist es möglich, RPA-Roboter in verschiedenen Modellierungssprachen zu erstellen. Dadurch werden Prozessexperten sowie Low-Code-Developer in die Lage versetzt, direkt in ihrer gewohnten Modellierungsumgebung die Automatisierungen zu programmieren.

Inhaltsverzeichnis

1. Einführung	1
2. Grundlagen	3
2.1. Einleitung zur Robotic Process Automation	3
2.2. Einleitung zu Modellierungssprachen	5
2.3. Verwandte Arbeiten	5
3. Anforderungen an eine RPA-Plattform	7
3.1. Dokumentation der Interviews	7
3.1.1. Interview: Prozessexperten aus der Verwaltung	8
3.1.2. Interview: Prozessexperten in IT-Projekten	8
3.1.3. Interview: RPA-Entwickler	8
3.1.4. Interview: Manager & CFO	9
3.2. Fazit	10
4. Analyse geeigneter Prozessmodellierungssprachen	11
4.1. Analyse des ausführbaren RobotFramework Codes	11
4.2. Analyse der Semantik ausgewählter Notationen	13
4.2.1. Analyse des Programmablaufplans	14
4.2.2. Analyse der BPMN	15
4.2.3. Analyse der EPK	16
5. Erläuterung der einheitlichen Speicherlösung	18
5.1. Konzeption der Architektur	18
5.2. Konzeption der Speicherlösung	20
6. Implementierung der einheitlichen Speicherlösung	23
6.1. Übersetzung der SSoT in eine visuelle Repräsentation	23
6.2. Übersetzung einer visuellen Repräsentation in die SSoT	24
6.3. Evaluation der Lösung	25
7. Zusammenfassung und Ausblick	27
Literatur	29
A. Anhang	31

Abbildungsverzeichnis

1.	Screenshot der Modellierungsoberfläche des viadee-Roboters	6
2.	Verbreitung von Notationen im deutschsprachigen Europa	13
3.	Beispielprozess in Flowchart-Syntax	14
4.	Beispielprozess in BPMN-Syntax	15
5.	Beispielprozess in EPK-Syntax	16
6.	Lokation der Parser ohne Wechsel zwischen den Darstellungsformen .	18
7.	Lokation der Parser ohne einheitliche Datenspeicherlösung	19
8.	Lokation der Parser mit einheitlicher Datenspeicherlösung	19
9.	Hierarchischer Aufbau der Datenspeicherlösung	21
10.	Fehlermeldung BPMN-Editor	26
11.	Fehlermeldung Code-Editor	26
12.	Beispielprozess in RobotFramework-Syntax	31
13.	Beispielprozess in RobotFramework-Syntax unter Verwendung der Keywords	32
14.	Auflistung der elementaren logischen Ablaufstrukturen	33

Tabellenverzeichnis

1.	Ausgangstabelle des Beispielprozesses	12
2.	Typen der Single Source of Truth	20

1. Einführung

Unternehmen stehen unter immensem Druck, ihre Prozesse und Abläufe zu automatisieren. Oftmals ist die Robotic Process Automation, die in den vergangenen Jahren ein beeindruckendes Wachstum erfährt [Gar], hierfür die geeignete Automatisierungsstrategie. RPA steigert die Produktivität der Abläufe durch den Einsatz sogenannter Softwareroboter. Diese Roboter übernehmen die Aktivitäten des Nutzers in Desktop- oder Webanwendungen, indem sie seine Interaktion mit den Oberflächen eigenständig ausführen. Da in den automatisierten Prozessen lediglich die menschlichen Softwareanwender durch den Menschen imitierende Roboter ersetzt werden, bleibt der Prozess unverändert. Dadurch entfällt der hohe Aufwand der Prozessintegration.

Durch das große Wachstum dieser Automatisierungstechnologie entstand eine Vielzahl von Plattformen, die das Erstellen der Softwareroboter ermöglichen. Jedoch haben die mit potenziellen Stakeholdern durchgeführten Interviews gezeigt, das Funktionen gewünscht werden, die keine auf dem Markt verfügbare Plattform abbildet. Wie im Fazit auf der Seite 10 detailliert beschrieben, dominierte in den Interviews der Wunsch nach einer RPA-Plattform, mit der Roboter durch verschiedene, standardisierte Modellierungssprachen wie zum Beispiel der BPMN, der Flussdiagramm-Syntax oder als Ereignisgesteuerte Prozesskette (EPK) erstellt werden können. Dies soll das Konfigurieren der Roboter für Low-Code-Entwickler¹ ermöglichen. Alle in der Branche bekannten Plattformen unterstützten lediglich das Modellieren von Robotern in deren proprietären Oberflächen; standardisierte Notationen können bislang nicht zum Erstellen einer Automatisierung verwendet werden.

In der ersten Forschungsfrage dieser Arbeit wird untersucht, ob sich Modellierungssprachen zur Erstellung von RPA-Robotern eignen. In der zweiten Forschungsfrage wird analysiert, wie das Erstellen desselben Roboters in verschiedenen Modellierungssprachen erfolgen kann, sodass jeder an der Automatisierung beteiligte Entwickler die ihm vertraute Modellierungssprache verwenden kann. Hierzu werden eine einheitliche Speicherlösung für RPA-Roboter sowie die zur Übersetzung in visuelle Repräsentationen notwendigen Parser vorgestellt.

¹Low-Code-Entwickler

Entwickler, die vor allem mit grafischen Programmoberflächen wie z. B. Blockeditoren oder Modellierungssprachen anstatt von textbasierter Programmierung arbeiten.

Nach einer Einführung in die RPA, einem Überblick über verwandte Arbeiten sowie der Dokumentation der Befragungen, wird im Kapitel 4 untersucht, ob sich Modellierungssprachen für die Entwicklung von RPA-Robotern eignen. Auf Grundlage dieser Betrachtungen ist im nachfolgenden Kapitel das Konzept der einheitlichen Speicherlösung beschrieben. Das Kapitel 6 erklärt, wie die beschriebenen Konzepte implementiert werden können. Abschließend werden die Implementierung evaluiert (Kap. 6.3), die wesentlichen Erkenntnisse zusammengefasst sowie ein Überblick über offene Forschungsfragen (Kap. 7) gegeben.

2. Grundlagen

Dieses Kapitel führt detailliert in die Robotic Process Automation ein und zeigt die Vorteile dieser Automatisierungsmethode auf. Zudem werden verwandte Arbeiten vorgestellt, die sich mit den Themen der Robotermodellierung in bekannten Notationen oder der einheitlichen Speicherung von Robotermustern befassen.

2.1. Einleitung zur Robotic Process Automation

In den vergangenen Jahrzehnten wurde das Workflow Management angewendet, um Arbeitsabläufe computergestützt zu automatisieren [AHH04]. Ein großes Problem bei der Einführung des Workflow Managements stellt die meist mangelhafte Akzeptanz der Mitarbeiter dar. Ebenso lassen sich vollständige Automatisierungen nur schwer in bestehende Systeme integrieren. Um diesen Problemen vorzubeugen, wird das Workflow Management seit einigen Jahren um das Business Process Management (BPM) ergänzt [Aal21]. BPM legt den Schwerpunkt auf die von den Mitarbeitern ausgeführten Aufgaben und ermöglicht durch die verwendete Notation BPMN eine vereinfachte Analyse und Optimierung des Prozesses. Da die Mitarbeitenden bei den Automatisierungen nun stärker im Fokus stehen, steigt die Akzeptanz derer bei der Einführung der Automatisierungslösung. Jedoch wird der BPMN vorgeworfen, durch den verstärkten Einsatz der Modellierung den zu automatisierenden Prozess aus dem Fokus zu verlieren. „Schlussendlich ist das Ziel nicht die Modellierung, sondern die Verbesserung des existierenden Prozesses“ [AHH04].

Um dem meist kostspieligen BPM mit den genannten Nachteilen entgegenzuwirken, wird seit einigen Jahren die Robotic Process Automation entwickelt. Sie wurde anfangs von Wirtschaftsunternehmen vorangetrieben und gerät nun seit vielen Jahren in den Fokus der Wissenschaft [Aal]. Ziel der RPA ist es, die vom Nutzer in Drittanwendungen ausgeführten Schritte zu erkennen, aufzuzeichnen und nachfolgend automatisiert auszuführen. Dabei interagiert der sogenannte Softwareroboter analog zum Nutzer mit der grafischen Oberfläche der Software. Da RPA direkt auf dieser Anwendungsebene arbeitet, fällt der Integrationsaufwand im Vergleich zu anderen Automatisierungsmethoden deutlich geringer aus. Es kann weiterhin die bestehende Software genutzt werden, es müssen keine offenen Schnittstellen entwickelt oder gar neue Hardware angeschafft werden. Zwingend notwendig sind jedoch eindeutig

definierte Aufgabenfolgen. So lassen sich mittels RPA zum Beispiel Formulare ausfüllen, Daten verschieben, Mails versenden aber auch Informationen von Webseiten verarbeiten.

RPA ist eine weltweit bekannte Technologie und wird bereits in zwei Dritteln der deutschen Unternehmen verwendet [ISG] [PwC]. Zudem ist die RPA mit 63 % Wachstum in 2018 das am schnellsten wachsende Segment der Softwareentwicklung [?]. Ebenso zeichnet sich die Technologie durch eine schnelle Kapitalrendite aus, die vor allem durch den Bottom-Up Ansatz begründet wird. „Oftmals werden die vollständigen Prozesse betrachtet, jedoch nur kleinere Gruppen von Aufgaben - die sogenannten »Quick-wins«- direkt automatisiert [?].“ Dies ist nur dadurch möglich, da die Roboter „Hand-in-Hand“ mit den Anwendern arbeiten.

Wie eingangs beschrieben, unterstützt BPM das Workflow Management bei der Prozessoptimierung. Zur Verbesserung der bestehenden Prozesse wird im Vorfeld einer Robotic Process Automation das Process Mining verwendet. Process Mining analysiert Geschäftsprozesse durch das Verarbeiten und Korrelieren von Datenströmen. Es erkennt Muster in den Prozessdaten und hilft somit das Verständnis der Prozesse zu verbessern. Dadurch lassen sich Möglichkeiten für die sinnvolle Einbindung von RPA erkennen. Vor der Implementierung einer RPA-Lösung empfiehlt sich eine Analyse des umzusetzenden Prozesses mittels Process Mining, da andernfalls der Prozess zwar automatisiert, jedoch vorher nicht optimiert wird.

Bekannte RPA-Plattformen entwickeln unter anderem die Firmen UiPath², blueprism³ und AutomationAnywhere⁴. Die Firma Microsoft stellt mit Power Automate Desktop seit März 2021 ihre eigene kostenlose RPA-Plattform zur Verfügung. Einige der Plattformen - wie zum Beispiel UiPath - besitzen ein ausgeprägtes Recoding-Feature, mit dem die zu automatisierenden Aufgaben direkt aufgenommen werden können. Andere hingegen lassen sich einfacher bedienen oder unterstützen eine Vielzahl an Drittanwendungen.

²<https://www.uipath.com/de/> (Abgerufen 12. Juni 2021)

³<https://www.blueprism.com/de/> (Abgerufen 12. Juni 2021)

⁴<https://www.automationanywhere.com/de/> (Abgerufen 12. Juni 2021)

2.2. Einleitung zu Modellierungssprachen

Um Organisationsstrukturen, Software aber auch Prozesse zu skizzieren, werden zur Beschreibung der Sachverhalte visuelle Modellierungssprachen verwendet [Win00]. „Neben der Unterstützung bei der Erfassung und Dokumentation sind diese grafischen und textuellen Beschreibungsmittel auch wesentliche Hilfsmittel zur Kommunikation bei der (Weiter-) Entwicklung von Organisationen und Softwaresystemen“, beschreibt A. Winter den Nutzen von Modellierungssprachen treffend. Modellierungssprachen unterscheiden sich in dem zur Verfügung gestellten „Vokabular und der Grammatik“, mit der Sachverhalte der menschlich wahrgenommenen Realität in Modellen abgebildet werden können [FOSS18].

In der durch die Object Management Group (OMG) standardisierten Unified Modeling Language (UML) ist eine Sammlung von Notationen definiert. Die UML umfasst unter anderem das in der Objektmodellierung bekannte Klassendiagramm, das für die Funktionsmodellierung genutzte Use Case Diagramm oder auch das Komponentendiagramm zur Beschreibung von IT-Systemstrukturen. Für dieses Werk von Relevanz sind die Notationen zur Prozessmodellierung. Diese Sprachen dienen der Darstellung von Interaktionsschritten innerhalb eines Prozesses und eignen sich daher unter Umständen auch zur Modellierung von RPA-Robotern. Bekannte Vertreter sind hierbei das Datenfluss-orientierte Flussdiagramm, die Kontrollfluss-orientierten Sprachen wie die BPMN, das Petri-Netz oder auch die ereignisgesteuerte Prozesskette [Lob].

2.3. Verwandte Arbeiten

In der bestehenden Literatur sind nur wenige Ansätze von Untersuchungen oder Prototypen für die Erstellung von RPA-Robotern in Modellierungssprachen zu finden. Dies ist sehr verwunderlich, da zahlreiche Artikel [AR17][FLL19] die Wichtigkeit der Dokumentation und Prozessanalyse im Zusammenspiel mit RPA erwähnen.

Eine interessante Arbeit ist womöglich die Abschlussarbeit des Studenten Johann Schäfer der FH Münster [Via]. Leider erhielt ich auch nach mehreren Kontaktversuchen keine Antwort auf die Anfrage, die Abschlussarbeit oder vielleicht sogar das Softwareprojekt betrachten zu dürfen. Daher war eine qualitative Analyse sowie eine aufbauende Forschung auf den Erkenntnissen der Arbeit des Studenten nicht möglich.

Der Autor beschreibt eine Weiterentwicklung des `bpmn-io` BPMN-Modelers, die sich zum Erstellen von RPA-Robotern eignen soll. Wie in Abbildung 1 zu sehen, ist es möglich, mit der entstandenen Plattform jeder BPMN Service Task eine RPA-Aufgabe zuzuweisen. In den grünen Boxen unterhalb der Task ist es möglich, der Aufgabe Parameter zu übergeben. Zudem wird ein aufklappbarer Subprozess verwendet, um auch kleinteilige Automatisierungsschritte übersichtlich darzustellen. Im Hauptprozess (oben dargestellt) ist durch die Annotation direkt erkennlich, welche Teile des Prozesses mittels RPA automatisiert werden.

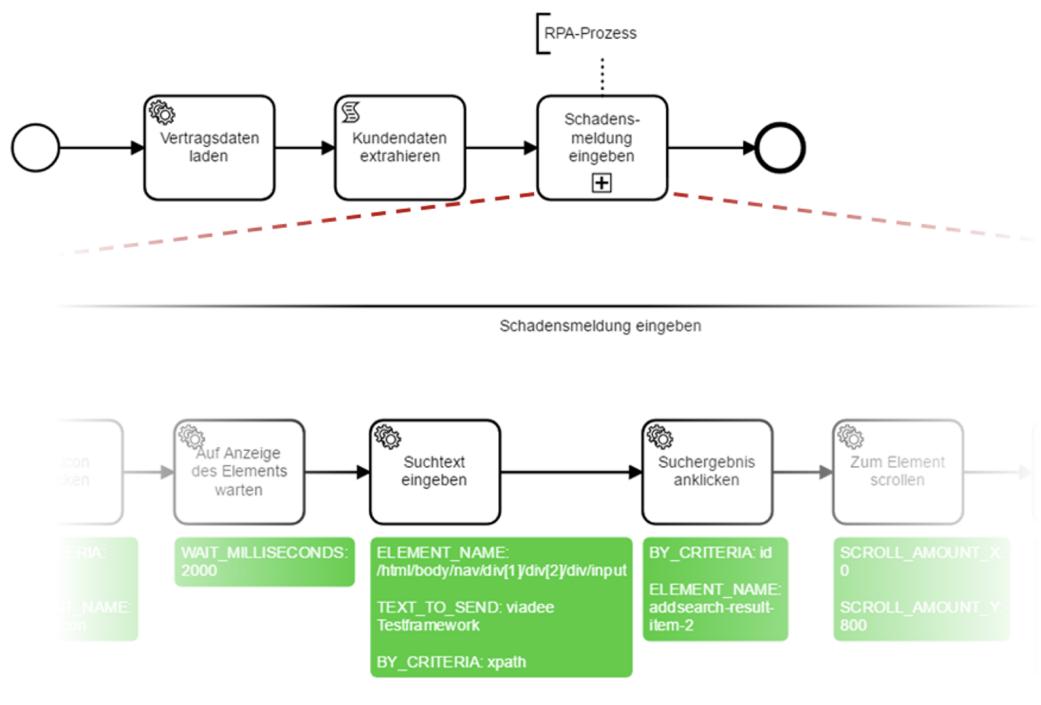


Abbildung 1: Screenshot der Modellierungsoberfläche des viadee-Roboters

3. Anforderungen an eine RPA-Plattform

UiPath veröffentlichte 2018 eine Auflistung [AV] von sieben Anforderungen, die eine moderne RPA-Plattform erfüllen sollte. Da die Veröffentlichung bereits einige Jahre zurückliegt und die Entwicklung der RPA-Plattformen stetig voranschreitet, wurden zu Beginn des Projektes verschiedene Stakeholder befragt. Ziel der elf Interviews war es, die Anforderungen an eine neue RPA-Plattform zu definieren und mit den Ergebnissen der UiPath-Studie abzugleichen.

In der nachfolgenden Zusammenfassung werden die für diese Arbeit relevanten Informationen zur Interaktion der Stakeholder mit den Modellierungsoberflächen hervorgehoben.

3.1. Dokumentation der Interviews

In den Interviews wurden neben Lehrstuhlmitarbeitenden, die unter anderem an Themen rund um RPA forschen, Prozessexperten, vier RPA-Entwickler⁵ sowie ein Manager befragt. Als Prozessexperten wurden aus der Verwaltung des Hasso-Plattner-Instituts eine Sekretärin zweier Fachgruppen sowie die Referentin für Forschung und Lehre interviewt. Als IT-Prozessexperte wurde ein Experte für Vulnerability Management des Cyber Defence Centers der Deutschen Telekom befragt, der seit über einem Jahr mit einem externen Team an softwaregestützten Prozessautomatisierungen arbeitet. Zudem wurde der General Manager und Chief Financial Officer (CFO) von Thermondo interviewt.

⁵RPA-Entwickler

haben auch beratende Funktionen, weshalb die Begriffe RPA-Consultant und RPA-Developer von nun an synonym verwendet werden.

3.1.1. Interview: Prozessexperten aus der Verwaltung

Die Prozessexperten der Verwaltung betonten, dass es in ihrem Büroalltag einen großen Bedarf für die Automatisierung und Digitalisierung von Prozessen gibt. Die tägliche Arbeit ist zwischen einem und zwei Dritteln von monotonen Verwaltungsaufgaben dominiert, die den Angestellten weder Spaß bereitet, noch sie im gewünschten Umfang bei der Arbeit effektiv voranbringt. Es wurden zahlreiche Prozesse vorge stellt, die wiederkehrend und auch fehleranfällig sind. Leider sind viele dieser Pro zesse nur teilweise digitalisiert, weshalb eine End-To-End-Automatisierung mittels RPA vorab eine Digitalisierung der Prozesse erfordert.

3.1.2. Interview: Prozessexperten in IT-Projekten

Der Cyber Defence Experte arbeitet seit sechs Monaten als Prozessexperte mit zwei externen Entwicklern zusammen, die für ihn alltägliche Routineaufgaben automatisieren. Er bemängelte, dass der Erstellungsprozess der Roboter sehr viel Zeit in Anspruch nimmt. Zu Beginn jedes Automatisierungsprojektes erläutert er den Entwicklern und Consultants welche Prozesse automatisiert werden sollen. Auf dieser Grundlage wird mit allen Beteiligten ein Programmablaufplan erstellt. Im Anschluss daran entwickeln die Consultants die Automatisierung mit ihrer Plattform und präsentieren den Roboter dem Kunden. Dieser hat abschließend jedoch keinen Zugriff auf die Automatisierung, was er bemängelt. Ebenso würde der Prozessexperte gern kleine Änderungen an den fertigen Automatisierungen in einem leicht zu bedienen Interface direkt vornehmen können. Auch dies ist mit den derzeit bestehenden Lösungen nicht möglich.

3.1.3. Interview: RPA-Entwickler

In umfangreichen Interviews wurden vier RPA-Developerbefragt, die mehrjährige Erfahrung im Bereich der RPA besitzen.

Die Entwickler arbeiten seit einigen Jahren mit der Software des Marktführers Ui-Path. Der Prozess zur Erstellung eines Softwareroboters beginnt mit der Prozessvorstellung durch den Kunden. Im Nachgang des Erstgespräches werden die Prozesse detailliert gezeichnet. Hierzu wird von den Befragten ausschließlich die Modellie

rungssprache BPMN verwendet. Auf Grundlage der Prozessskizze wird mögliches Potenzial zur Optimierung der Abläufe analysiert, bevor es zur finalen Besprechung mit dem Kunden erfolgt. Danach beginnt die Implementierung des Roboters in UiPath. Hierbei merkten beide Entwickler an, dass das gedankliche Übersetzen eines in BPMN gezeichneten Prozesses in die proprietäre UiPath-Syntax unnötig kompliziert ist. Aus ihrer Sicht sollte es auch möglich sein, einen RPA-Roboter direkt in der BPMN-Syntax zu erstellen. Einer der Entwickler sagte zudem, dass nach Fertigstellung des Roboters dieser noch dokumentiert werden muss. Dafür eignet sich seiner Meinung nach BPMN als Dokumentationssprache hervorragend. Da der final implementierte Roboter oftmals von der ursprünglichen Skizze abweicht, muss dieser zu Dokumentationszwecken erneut in BPMN skizziert werden. Gewünscht wurde auch eine Dokumentationshilfe, die die verwendeten RPA-Befehle direkt in die Dokumentation einfließen lässt. Solch eine Funktion sucht man bisher auf dem Markt vergeblich.

Ebenso merkten zwei Entwickler an, dass die Oberfläche von UiPath sehr komplex und benutzerunfreundlich ist. Zwei Entwickler suchen daher seit langer Zeit nach einem Tool, dass sich leichter bedienen lässt. Leider umfassen diese Tools meist nicht alle benötigten Funktionen, weshalb schlussendlich doch wieder auf UiPath zurückgegriffen werden muss.

Weiterhin wurde von einem Entwickler ein "Marktplatz" für Roboter gewünscht, der es ermöglichen sollte, Roboter innerhalb der Organisation oder auch in einer Open-Source Community zu teilen. Zudem sollte es eine Administrationsoberfläche geben, von der aus zentral Aktualisierungen eingespielt werden können.

3.1.4. Interview: Manager & CFO

Seit mehreren Jahren ist der Manager bestrebt, analoge Prozesse im Heizungs- und Handwerkerwesen zu digitalisieren. Er hofft, dass seine Firma durch eine höhere Anzahl automatisierter Prozesse in Zukunft besser skalieren kann. Seine wichtigste Feststellung aus den vergangenen Jahren war, dass man in Deutschland meilenweit von vollständig „automatisierbaren“ Prozessen entfernt ist. Der Grund hierfür sei, dass sich softwarebasierte Prozesse aufgrund veralteter Software und somit fehlender Schnittstellen (APIs) nicht automatisieren lassen. Er zeigte sich begeistert von der für ihn unbekannten Zukunftstechnologie RPA und wünschte sich eine Plattform,

mit der auch Prozessexperten Roboter erstellen können. Der Manager betonte, dass viele Bestrebungen nach verstärkter Automatisierung nicht nur an fehlender Digitalisierung scheitern. „Entwickler und Prozessberater sind schlichtweg zu teuer“, verriet der CFO.

3.2. Fazit

Die vielfältigen Antworten der Befragten waren grundlegend für die Definition der Anforderungen an die neue Plattform. Zusammenfassend lässt sich festhalten, dass in allen Branchen, die die Befragten repräsentieren, ein ausgesprochen großes Interesse an den Möglichkeiten der Automatisierung bestand, um die alltägliche Arbeit zu erleichtern und zu effektivieren. Die Mitarbeitenden der Verwaltung sagten uns, dass bei der Entwicklung der Plattform in jedem Fall ein besonderes Augenmerk auf die alltagstaugliche Bedienung und schnelle Erlernbarkeit gelegt werden sollte. Die RPA-Developer wünschten sich vor allem eine leicht zu bedienende RPA-Plattform, mit der bestenfalls direkt in BPMN Roboter skizziert, Automatisierungen erstellt und abschließend die Prozesse dokumentiert werden können. Thermondos CFO verdeutlichte, dass mit einer guten RPA-Plattform auch Prozessexperten Roboter erstellen können sollen. „Die Kosten für RPA-Consultants sind für viele Firmen eine Herausforderung“, so der Manager.

Die Auswertung der Interviews zeigt, dass sich der Großteil der Anforderungen aus der UiPath-Studie in den aktuellen Wünschen der Entwickler widerspiegelt. Die Studie erachtet es als zwingend notwendig, dass die Roboter durch die Mitarbeitenden der Fachabteilungen erstellt werden können. Ebenso erwähnt die Veröffentlichung die Anforderung eines Marktplatzes für Roboter, mit der die „globale Anwender-Community“ von den fertiggestellten Automatisierungen profitiert. Die Auflistung von UiPath bestätigt die Notwendigkeit eines Administrations-Interfaces und beschreibt zudem noch die Management-Konsole, die es den Mitarbeitern erleichtern soll, die Automatisierungen zu starten, zu kontrollieren und zu validieren. Einzig die Nutzung von Künstlicher Intelligenz, die als eine Anforderung von UiPath beschrieben wurde, konnte nicht aus den Interviews abgeleitet werden.

4. Analyse geeigneter Prozessmodellierungssprachen

„Von der Wahl der Modellierungssprache hängt also ab, was in einem Modell überhaupt abgebildet werden kann, und was nicht sichtbar werden kann, weil die Modellierungssprache keine Konzepte dafür anbietet.“ [FOSS18] Die Interviews ergaben die Notwendigkeit der Funktion, RPA-Roboter in Modellierungssprachen zu erstellen. Daher wird nachfolgend untersucht, welche Notationen sich für die Implementierung von RPA-Robotern eignen.

4.1. Analyse des ausführbaren RobotFramework Codes

Betrachten wir vor der Analyse der Modellierungssprachen die Syntax der Programmiersprache RobotFramework⁶, um im Anschluss bewerten zu können, welche Programmierkonzepte von den Notationen unterstützt werden müssen. RobotFramework wurde für die Ausführung der Automatisierungen gemeinsam mit den Bibliotheken des RpaFrameworks⁷ verwendet.

RobotFramework ist ein quellenoffenes Framework, das für die Testautomatisierung und RPA verwendet werden kann. Es bietet einen leicht lesbaren Code, mit dem auch Einsteiger Automatisierungen erstellen können. Das Framework wird aktiv weiterentwickelt und von zahlreichen Unternehmen wie beispielsweise DB Schenker, Capgemini, Kuka oder auch der Deutschen Post verwendet. Alle zugrunde liegenden Bibliotheken sind in Java oder Python implementiert und können daher bei Bedarf erweitert werden.⁸

Die Bibliotheken des RpaFrameworks stellen die RPA-Aufrufe zur Verfügung. Die zugrunde liegende Programmierlogik wird in der Syntax der Test-Entwicklungssprache RobotFramework geschrieben. Daher wird sich im Verlauf der Arbeit vor allem auf diese Syntax bezogen.

RPA-Roboter beschreiben Nutzerinteraktionen in verschiedenen Desktopanwendungen oder Browsern. Da es sich bei den Automatisierungen um ausführbare Programme handelt, besitzen diese die typischen Elemente einer jeden Programmiersprache.

⁶www.robotframework.org (Abgerufen 19. Juni 2021)

⁷www.rpaframework.org (Abgerufen 19. Juni 2021)

⁸sämtliche Informationen entnommen aus:

<https://robotframework.org/#introduction> (Abgerufen 19. Juni 2021)

Somit finden sich in RobotFramework sequenzielle Anweisungen, Verzweigungen sowie terminierte Wiederholungen (implementiert als **FOR**- oder **WHILE**-Schleife). Zudem werden in RobotFramework Variablen unterstützt. Oftmals werden diese genutzt, um das Ergebnis einer RPA-Aktivität (zum Beispiel «Microsoft Excel» → »Get Cell Value«) zwischenspeichern und dann als Eingabeparameter einer anderen Aktivität anzugeben. Die Konzepte von Try-Catch Blöcken sind aktuell noch nicht vollständig implementiert und werden daher in dieser Arbeit nicht weiter betrachtet.

Anhand eines Beispiel-Roboters, der den elektronischen Versand eines Mahnbescheides automatisiert, werden nachfolgend die theoretischen Grundlagen erläutert. Dem Roboter zugrunde liegt eine EXCEL-Tabelle, die täglich von den Mitarbeitenden der Verwaltung gefüllt wird und den Vor- und Nachnamen des Kunden, die Mail-Adresse sowie den Mahnungszähler 1 oder 2 beinhaltet. Die letzte Zahl gibt an, die wievielte Zahlungsaufforderung an den Kunden gesendet werden soll. Der Roboter öffnet jede Nacht die erstellte EXCEL-Tabelle und geht Zeile für Zeile die Einträge der Tabelle durch.⁹ Er liest dafür jede Zeile einzeln ein und prüft im Anschluss den Mahnungszähler. Abhängig von der eingetragenen Zahl wird er nun eine Mail an den Kunden erstellen und mit dem entsprechenden Mahnbescheid als Anlage versehen. Sobald der Roboter am Ende der Tabelle angekommen ist, terminiert die Automatisierung und das Mail-Programm schließt sich.

Tabelle 1: Ausgangstabelle des Beispielprozesses

Lastname	Firstname	Mail	Counter
Stella	Patricia	p.stella@outlook.com	1
Matthäus	Clara	familie-matthaeus@gmx.com	2
Zenzi	Mathis	m.zenzi@web.de	1
Reinhold	Cornelia	conny.reinhold@t-online.de	1

Im Anhang ist der ausführbare Prozess in der Syntax des RobotFrameworks dargestellt (Abbildung 12). Dabei wurde an einigen Stellen auf die Verwendung von Test-Case Bezeichnungen, die im übrigen Code rosa eingefärbt sind, verzichtet. Diese Einschränkung war notwendig, da jeder Testcase einen eigenen Variablen-Scope bildet, sodass die lokalen definierten Variablen in darauffolgenden Anschnitten nicht wiederverwendet werden können.

⁹Das automatische Starten des Roboters wird aus Komplexitätsgründen nicht betrachtet.

4.2. Analyse der Semantik ausgewählter Notationen

Um einen Roboter vollumfänglich in einer Modellierungssprache erstellen zu können, muss diese mindestens genau so mächtig wie die verwendete Sprache der Automatisierung (in unserem Fall RobotFramework) sein. Diese Mächtigkeit findet sich vor allem in Modellierungssprachen, die es ermöglichen, Softwareprogramme oder Geschäftsprozesse abzubilden. In diesem Abschnitt werden die sehr verständlichen und aussagekräftigen Flussdiagramme [Wil07] sowie die BPMN, der meist eingesetzte Standard zur Abbildung von Geschäftsprozessen [FOSS18], analysiert. Zudem werden die laut einer Studie zur Verbreitung von Notationen im deutschsprachigen Europa (Abbildung 2 [Lob]) sehr bekannten Ereignisgesteuerten Prozessketten (EPK) betrachtet. Diese sind jedoch kaum spezifiziert und werden daher außerhalb des deutschsprachigen Raumes nur selten verwendet [MCKL11].

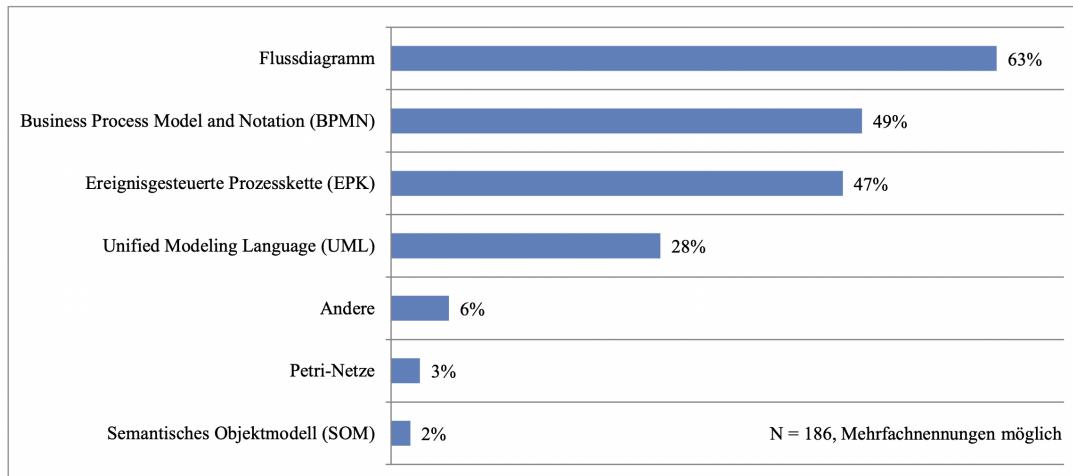


Abbildung 2: Verbreitung von Notationen im deutschsprachigen Europa

Der finale Ausführungscode kann auch als eine Modellierungssprache angesehen werden, da er trivialerweise alle notwendigen Programmierkonzepte unterstützt. Hier soll jedoch die Betrachtung der oben genannten Sprachen genügen.

4.2.1. Analyse des Programmablaufplans

Der Programmablaufplan (auch oft als Flussdiagramm bezeichnet) ist in der ISO 5807 sowie DIN 66001 beschrieben. Wie in [Her84] erläutert, dient der Programmablaufplan zur Darstellung der „Grundstruktur der Datenverarbeitung“. Es stehen so genannte „Sinnbilder“ zum Abbilden der bekannten Konzepte der Sequenz, Verzweigung sowie Wiederholung zur Verfügung. Dariüber hinaus sind ebenso im Standard Kommentarfelder definiert, die an jedes Sinnbild angehangen werden können. Lediglich für Variablen existiert keine vordefinierte Syntax. Hierfür könnte jedoch das Sinnbild des Parallelogramms genutzt werden, das laut Standard Eingaben und Ausgaben definiert.

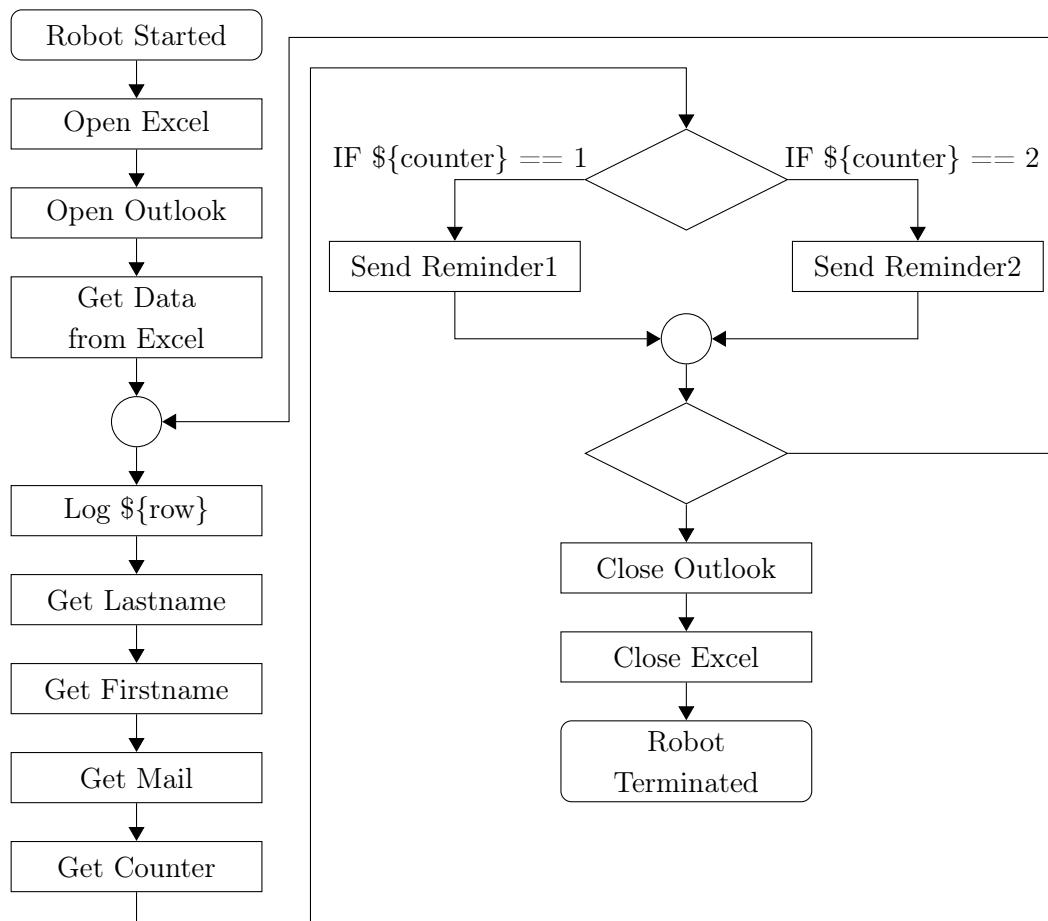


Abbildung 3: Beispielprozess in Flowchart-Syntax

Wie in der Abbildung 3 zu sehen, eignen sich Flussdiagramme (engl. Flowchart) sehr gut für die Darstellung eines Softwareroboters. Der Arbeit ist eine Auflistung der elementaren logischen Ablaufstrukturen aus dem Werk von E. Hering [Her84] beigefügt (Abbildung 14).

4.2.2. Analyse der BPMN

Wie in Abbildung 4 gezeigt, können sequenzielle Anweisungen in BPMN als Folge von Aktivitäten dargestellt werden. Verzweigungen lassen sich als XOR-Split mit entsprechenden Fallbedingungen abbilden. Ebenso können mit einem XOR-Split Wiederholungen implementiert werden. Sofern Variablen auch in der visuellen Repräsentation sichtbar werden sollen, lassen sich diese durch Datenobjekte darstellen.

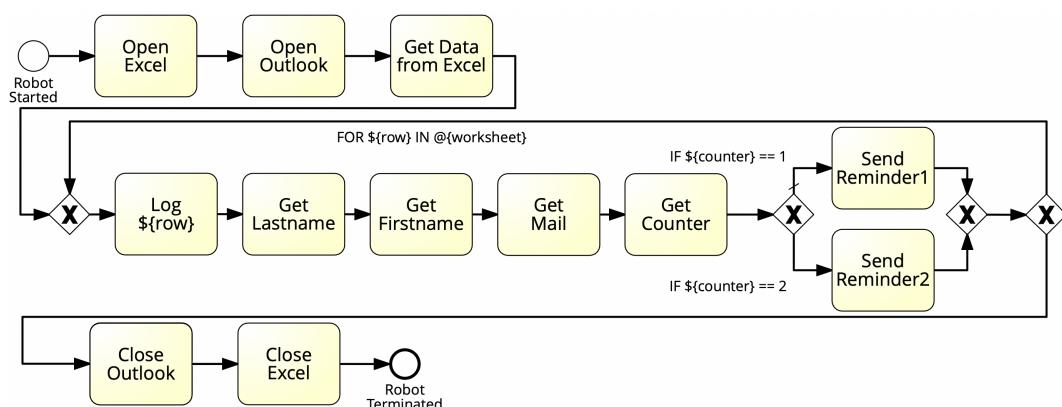


Abbildung 4: Beispielprozess in BPMN-Syntax

Ursprünglich ist die Prozessmodellierungssprache BPMN nicht dafür konzipiert worden, Softwareabläufe darzustellen. Wie soeben gezeigt, eignet sich die Notation neben der Prozessmodellierung auch für die Implementierung von RPA-Robotern. Zudem bietet sie bereits im Standard die Möglichkeit alle Aktivitäten zu dokumentieren. Dieses Feature wurde von den RPA-Entwicklern ausdrücklich gewünscht und gilt als ein zentraler Vorteil der BPMN und des Programmablaufplans gegenüber anderen Prozessmodellierungssprachen.

4.2.3. Analyse der EPK

Die Ereignisgesteuerte Prozesskette wurde an der Universität des Saarlandes zusammen mit der SAP AG zur Dokumentation von Geschäftsprozessen entwickelt [NR02]. Die EPK besitzt als Kernelemente das Ereignis, das einen Zustand im Prozess abbildet sowie die Funktion, die eine Aktion oder Aufgabe beschreibt. Jeder Prozess startet und endet mit einem Ereignis. Zwischen beiden Ereignissen können sequentielle Anweisungen durch aneinander gereihte Funktionsaufrufe dargestellt werden. Durch Einsatz des XOR-Konnektors lassen sich analog zur BPMN (Abbildung 5) Verzweigungen sowie Schleifen darstellen. Hierbei ist zu beachten, dass nach jeder XOR-Verzweigung zwingend ein Ereignis folgen muss, um den vorliegenden Zustand zu beschreiben. Diese Ereignis-Elemente dienen in unserem Fall zur Angabe der Fallbedingungen einer Verzweigung oder zur Darstellung der Schleifenabbruchbedingung.

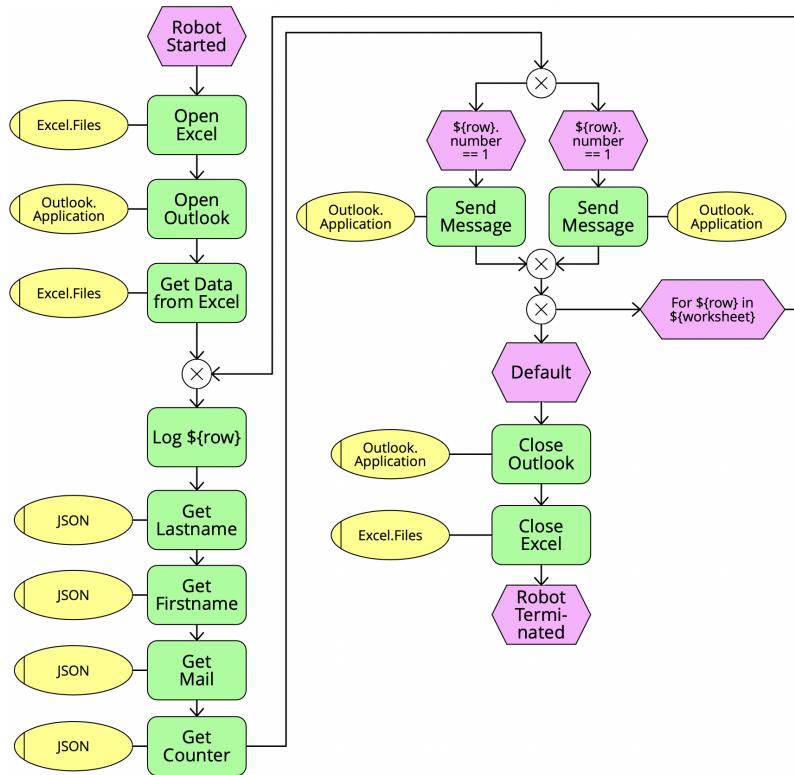


Abbildung 5: Beispielprozess in EPK-Syntax

Wie beschrieben, lassen sich auch mit EPKs RPA-Roboter darstellen. Ähnlich zur BPMN können dem Diagramm Datenobjekte hinzugefügt und dadurch Variablen repräsentiert werden. Darüber hinaus lässt das „Organisationseinheit“-Element für die Darstellung der zum Ausführen der RPA-Aufgabe genutzten Anwendung (z.B. `Excel.Application`) verwenden. Dieses zusätzliche Element kann in der jedoch auch Visualisierung kann jedoch entfernt werden, da zu jeder Funktion die RPA-Anwendung gesondert im Seitenmenü spezifiziert werden muss.

5. Erläuterung der einheitlichen Speicherlösung

Die Interviews des Kapitels 3 ergaben, dass eine neue Plattform gewünscht wird, bei der das Wechseln zwischen verschiedenen visuellen Repräsentationen der Roboter möglich ist. Dadurch können sowohl Low-Code-Developer als auch professionelle Entwickler in ihrer präferierten Umgebung Automatisierungen erstellen.

Nachfolgend wird erläutert, mit welcher Architektur sich diese Funktionalität implementieren lässt und warum hierfür eine einheitliche Speicherlösung (Single Source of Truth) benötigt wird. Im Anschluss werden die Anforderungen an die SSoT dargestellt.

5.1. Konzeption der Architektur

Um die visuelle Repräsentation eines Roboters in den ausführbaren Code der RobotFramework Syntax zu übersetzen, werden die eingangs beschriebenen Parser benötigt. Sofern ausschließlich Robotermodelle in ausführbaren Code übersetzt werden sollen und somit kein Wechsel der Modellierungssprache bei der Erstellung eines Roboters möglich ist, genügt die in Abbildung 6 gezeigte Architektur. Hierbei wird für jeden Roboter die visuelle Repräsentation um die benötigten RPA-Eigenschaften angereichert und gespeichert. Im Beispiel von bpmn-js ist das Speicherformat ein XML, in dem jede BPMN-Aktivität um die Tags `<rpaApplication>` und `<rpaTask>` erweitert werden kann. Soll nun der Roboter ausgeführt werden, wird die aktuelle Version des XML-Files in den ausführbaren Code geparsst. Dadurch wird für jedes unterstützte Interface genau ein Parser benötigt.

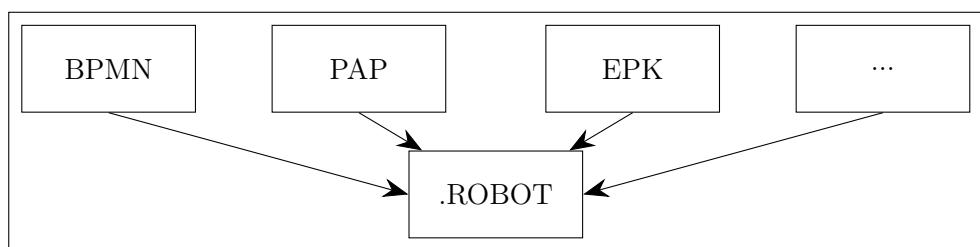


Abbildung 6: Lokation der Parser ohne Wechsel zwischen den Darstellungsformen

Um denselben Roboter in verschiedenen Notationen erstellen zu können, bedarf es in einer trivialen Lösung zwischen jedem Paar von Interfaces zweier Parser. Diese Variante der Implementierung ist in der Abbildung 7 dargestellt. Bei dieser und der nachfolgenden Lösung ist es zudem möglich, direkt in dem auszuführenden Code Änderungen vorzunehmen und diese im Anschluss auch in den Notationen zu sehen. Problematisch ist jedoch, dass bei dieser Architektur exponentiell viele Parser in Abhängigkeit der angebotenen Interfaces benötigt werden. Zudem müsste sich auf eine Repräsentation geeinigt werden, in der der Roboter persistent gespeichert wird.

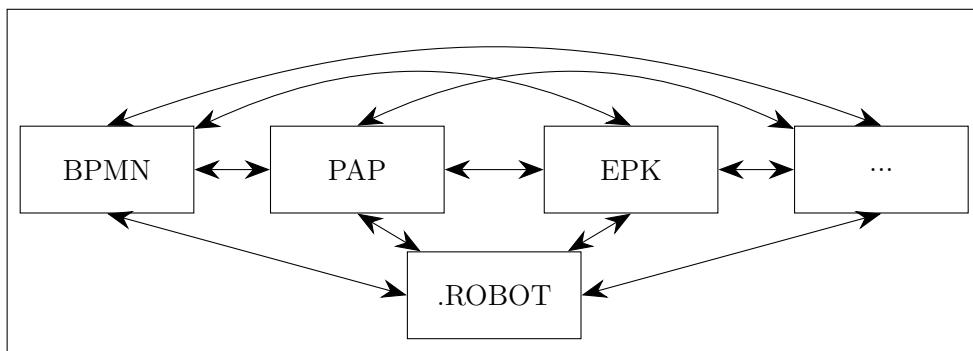


Abbildung 7: Lokation der Parser ohne einheitliche Datenspeicherlösung

Um nur linear viele Parser zu benötigen, stellt diese Arbeit eine Architektur vor, die auf einer Single Source of Truth basiert. Diese SSoT dient als einheitliches Speicherformat der Roboter. Aus ihr werden alle visuellen Repräsentationen erstellt und ebenso resultieren alle Änderungen in den grafischen Editoren in der SSoT. Zur Realisierung dieses Konzeptes werden pro Notation zwei Parser benötigt (Abbildung 8).

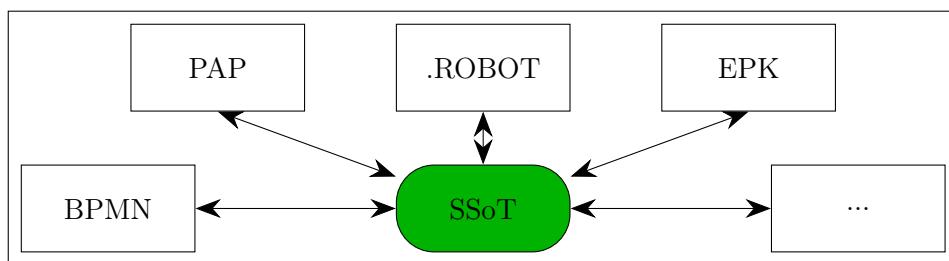


Abbildung 8: Lokation der Parser mit einheitlicher Datenspeicherlösung

5.2. Konzeption der Speicherlösung

Betrachten wir nun das Konzept hinter der einheitlichen Speicherlösung. Diese Datenstruktur speichert alle relevanten Daten des Roboters, sodass daraus der finale Ausführungscode sowie alle visuellen Repräsentationen erstellt (engl. „geparst“) werden können.

Ziel der Speicherlösung ist es, die visuellen Repräsentationen zu abstrahieren und nur die Semantik zu speichern. Dafür wird für jede Modellierungssprache die ihr zugrunde liegende Programmierlogik abgeleitet und nur diese gespeichert. Für die Programmierkonzepte werden nachfolgend Typen definiert, mit deren Hilfe sich alle Graphen abspeichern lassen (siehe Tabelle 2). Zusätzlich ist der Typ „**MARKER**“ definiert, der entweder Start oder Ende des Roboters beschreibt. Dadurch wird ermöglicht, die Start- und Endsymbole der BPMN, Flowcharts und EPKs - wie im Standard gewünscht - zu benennen.

Tabelle 2: Typen der Single Source of Truth

Programming Construct	SSoT-Type
Single Statement	INSTRUCTION
Branching	CASE
Loop	LOOP
Start / End of Robot	MARKER

Die SSoT besteht aus einem Header, der die Meta-Informationen des Roboters wie die **RobotId**, den Namen des Roboters und die Id des ersten Knotens im Graphen speichert. Darauf folgt das **ElementArray**, das alle Programmteile des Roboters enthält. Jedes Element des Element-Arrays besitzt ein Feld zur Angabe des Typs (**INSTRUCTION**, **CASE**, **LOOP** o. **MARKER**). Zudem wird für jedes Element der Name, eine Id, die Id der Vorgänger- und Nachfolger im Graphen sowie optional ein Dokumentationstext gespeichert.

Wie in Abbildung 9 zu sehen, kann jedes Element in Abhängigkeit des Typs um weitere Felder angereichert werden. Dazu zählen beispielsweise bei einer Anweisung (**INSTRUCTION**) die benötigten Informationen des RPA-Statements. Das Schleifen-Element (**LOOP**) speichert neben der Schleifenabbruchbedingung die Id's der Anfangs-

und Endknoten des Schleifenkörpers, um eine vollständige Rekonstruktion des Graphens zu ermöglichen. **MARKER** benötigen keine weiteren Attribute.

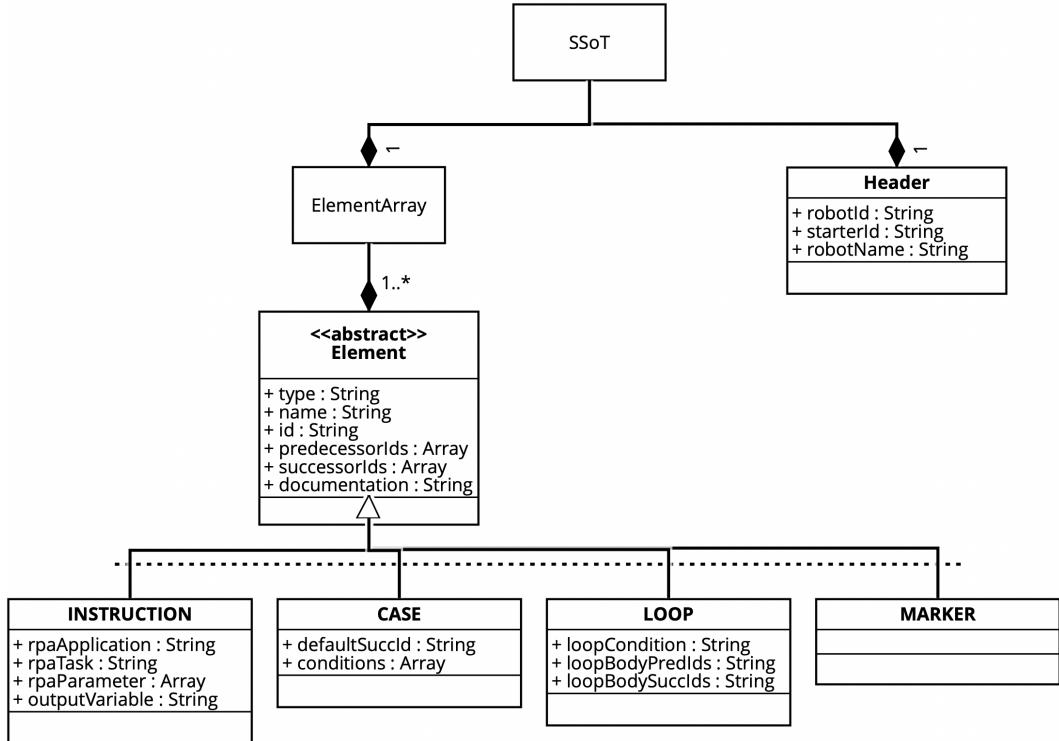


Abbildung 9: Hierarchischer Aufbau der Datenspeicherlösung

Der den Roboter beschreibende Graph lässt sich mit den Informationen zu Vorgänger- und Nachfolgerknoten eindeutig zusammensetzen. Somit kann aus der SSoT jede visuelle Repräsentation rekonstruiert werden. Es werden keine spezifischen Eigenschaften der Modellierungssprachen, wie zum Beispiel die absolute Position eines Knotens gespeichert. Daher müssen diese Werte auf Grundlage des Graphens errechnet werden, wodurch die Modelle sich bei jedem erneuten Rendering automatisch ausrichten und formatieren.

Die meisten Felder der Elemente werden mit Strings befüllt. Eine Besonderheit ist hierbei das **rpaParameter**-Array des **INSTRUCTION**-Elements. In diesem Array werden benötigte Eingabeparameter für die ausgewählte **rpaApplication/rpaTask**-Kombination angegeben. Somit kann dieses Array auch leer sein, sofern der Funktionsaufruf keine Parameter benötigt.

Jeder Parameter repräsentiert wiederum ein Objekt, in dem der Name des Parameters, ein Hinweistext, eine boolesche Variable (die aussagt, ob der Parameter zwingend angegeben werden muss), sowie die Position des Parameters aus der RpaFramework-Dokumentation angegeben wird. Diese Ganzzahl gibt in Zusammenhang mit den anderen Parametern an, in welcher Ordnung die Parameter in den auszuführenden Code zu parsen sind. Zudem werden für jeden Parameter der Typ wie beispielsweise **String** oder **Integer** sowie dessen Wert gespeichert.

6. Implementierung der einheitlichen Speicherlösung

Die nachfolgend beschriebene Implementierung ist als Proof of Concept im Open-Source-Projekt `ArkAutomate`¹⁰ umgesetzt. Die Entwicklung erfolgte im Rahmen des Abschlussarbeiten zugrunde liegenden Projektes. Das Projektteam implementierte die in dieser Arbeit beschriebene einheitliche Datenspeicherlösung sowie die Modellierungsfunktion als ein Kernfeature der quelloffenen RPA-Plattform. Die SSoT ist in diesem Projekt als verschachteltes JavaScript Object implementiert.

Betrachten wir nun das beidseitige Parsing zwischen der visuellen Repräsentation und der SSoT. Wie im vorangegangenen Kapitel beschrieben, werden zwischen jedem Interface und der SSoT zwei Parser benötigt. Ein Parser übersetzt das Modell in die SSoT und ein weiterer Parser erstellt aus der SSoT die visuelle Repräsentation.

Die Implementierung der Parser hängt stark von dem verwendeten Modellierungsmodul ab. Das im Proof of Concept verwendete npm-Paket `bpmn-js`¹¹ zeichnete sich durch seine umfangreichen Funktionen und Anpassungsmöglichkeiten des Modelers aus. Zur Modellierung von Flowcharts bietet sich eine Erweiterung des `react-flow-diagram` Paketes¹² an. Es lässt sich um die unterstützten Sinnbilder erweitern. Einen Modeler-Package zur Erstellung von Ereignisgesteuerten Prozessketten konnte nicht gefunden werden. Dies kann an der geringen Verbreitung der Notation außerhalb des deutschsprachigen Raumes liegen.

Nachfolgend werden die grundlegenden Konzepte der Parser an den im Proof of Concept verwendeten Notationen BPMN und dem finalen Ausführungscode beschrieben.

6.1. Übersetzung der SSoT in eine visuelle Repräsentation

Betrachten wir zuerst den Parser, der den RobotFramework Code erzeugt. Zu Beginn dieses Codes müssen die verwendeten RpaFramework Librarys aufgeführt werden (Abbildung 12). Hierfür genügt eine Iteration über das `ElementArray`, bei der alle verwendeten Applikationen gespeichert werden. Der Graph lässt sich mit einer rekur-

¹⁰https://github.com/bptlab/ark_automate (Abgerufen am 22. Juni 2021)

¹¹<https://www.npmjs.com/package/bpmn-js> (Abgerufen am 22. Juni 2021)

¹²<https://www.npmjs.com/package/react-flow-diagram> (Abgerufen am 22. Juni 2021)

siven Betrachtung des `ElementArrays` rekonstruieren, wodurch der RobotFramework Code zeilenweise wächst.

Komplizierter hingegen ist die Erstellung von Modellen aus der SSoT. Hierbei sind die verwendeten XMLs oft sehr komplex und erfordern eine Vielzahl an Parametern wie Positionen der Elemente und Verbindungen. Für das Paket `bpmn-js` existiert jedoch ein Command Line Interface (CLI)¹³, das die Ausrichtung der BPMN-Komponenten übernimmt. So genügt es, dem BPMN-Diagramm die Elemente in der sortierten Reihenfolge mittels `cli.append(element)` hinzuzufügen. Das anzulegende XML mit allen Positions値en wird durch das Interface erstellt.

6.2. Übersetzung einer visuellen Repräsentation in die SSoT

Zur einfacheren Speicherung der Informationen eines Roboters wird in der Implementierung der eigentlichen Graph getrennt von den Eigenschaften jedes Knotens gespeichert. Die Informationen zu `rpaApplication` und `rpaTask` werden zusammen mit einer Referenz auf das betreffende `INSTRUCTION`-Element in der `localAttributeStorage` abgelegt. In diesem einen Objekt der Session Storage werden somit direkt alle Änderungen an der RPA Task / Application-Kombination vermerkt. Das `localParameterStorage`-Objekt speichert zu jedem `INSTRUCTION`-Element ein Array aller Parameter mit den in Abbildung 9 vorgestellten Feldern. Die Seitenleiste (`ModelerSidebar`) bietet die Möglichkeit, alle Parameter eines Elements zu verändern. Die Änderungen werden direkt in den betreffenden Objekten der Session Storage geändert. Die `ModelerSidebar` ist modular aufgebaut und kann daher weiteren in Modellierungsinterfaces wiederverwendet werden.

Betrachten wir nun das Parsing - die eigentliche Erstellung der SSoT. Hierbei lassen sich für alle Parser dieser Richtung einige Gemeinsamkeiten festhalten. Jeder Parser erstellt zuerst den SSoT-Header mit den beschriebenen drei Feldern. Im Anschluss muss das `ElementArray` erstellt werden, sodass dann beide Teile zur SSoT zusammengefügt und gespeichert werden können.

Sofern die Daten in einer nicht strukturierten Form vorliegen, müssen sie vorab aufbereitet werden. Dieser Schritt geschieht beim RobotFramework Code durch zeilenweises Einlesen. Die Daten des `bpmn-js` liegen im XML-Format vor und müssen

¹³<https://www.npmjs.com/package/bpmn-js-cli> (Abgerufen am 22. Juni 2021)

daher ebenso vorab bearbeitet werden. Das XML lässt sich in der JavaScript Anwendung schwer verarbeiten, weshalb es in der Projektarbeit mit dem `xmljs2`-Paket¹⁴ in ein JavaScript Objekt gewandelt wurde. Auf dieser Grundlage können die Elemente durch eine einfache Iteration über die strukturierten Daten erstellt werden. Der BPMN-Parser muss das `bpmn-js`-XML um die Informationen der ausgelagerten Objekte aus der Session Storage anreichern.

6.3. Evaluation der Lösung

Die vorgestellte Lösung, die in ersten Teilen im Proof of Concept implementiert wurde, ermöglicht das Bearbeiten der Roboter in verschiedenen Modellierungssprachen. Die Parser, die aus der SSoT die visuelle Repräsentation des Roboters erstellen, ließen sich leicht implementieren. Das lag vor allem an dem durch Camunda bereitgestellten Interface sowie dem `bpmn-js` Modeler. Sofern diese Tools für eine Modellierungssprache - wie beispielsweise die EPKs - nicht zur Verfügung stehen, erhöht sich der Arbeitsaufwand zur Anbindung weiterer Interfaces erheblich. Aufwändig ist vor allem das korrekte Ausrichten der Elemente des Modells, was im Beispiel des BPMN-Editors das CLI übernahm.

Die Parser, die beim Speichern den Roboter in die SSoT übersetzen, implementieren sich vergleichsweise komplizierter. Um Fehler bei der Erstellung der SSoT zu vermeiden, ist eine komplexe Fehlerbehandlung notwendig, die unsaubere Eingaben des Codes abfängt. Gleiches gilt für den Parser, der das BPMN-Diagramm in die SSoT übersetzt. Sobald ein Fehler beim Übersetzen festgestellt wird, bricht der Parser ab und es wird keine SSoT an den Server gesendet. Bei beiden Parsern wird ein besonderer Fokus auf aussagekräftige Fehlermeldungen gelegt, die den Nutzern bei der Bedienung helfen. So wird der Entwickelnde unter anderem gewarnt, wenn das BPMN-Modell zwei Start-Ereignisse besitzt (Abbildung 10) oder wenn ein Schreibfehler im Code-Editor erkannt wird (Abbildung 11).

Das Proof of Concept unterstützt aktuell die Element-Typen `MARKER` und `INSTRUCTION`. Um weitere Typen zu unterstützen, ist eine Umstellung des RobotFramework Codes auf die Keywords-Syntax notwendig. Hierfür müssen die zwei betreffenden Parser angepasst werden. Die Keywords-Syntax ermöglicht das Zusammenfassen von Programmteilen ähnlich der Definition von Funktionen in der

¹⁴<https://www.npmjs.com/package/xmljs2> (Abgerufen am 22. Juni 2021)

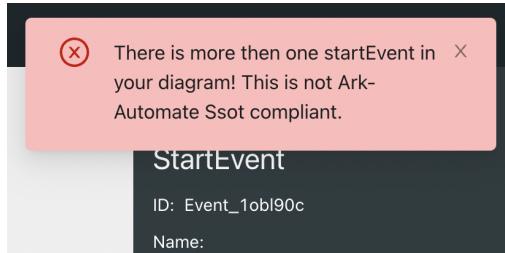


Abbildung 10: Fehlermeldung
BPMN-Editor

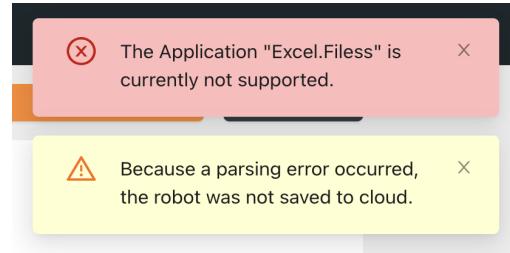


Abbildung 11: Fehlermeldung
Code-Editor

objektorientierten Programmierung. Diese Funktionen werden als Keywords bezeichnet und im Hauptprogramm semantisch aneinander gereiht (Abbildung 13). Sobald diese Umstellung erfolgt, lassen sich auch die beschriebenen Konzepte der Schleife und der Verzweigung implementieren. Mit der Weiterentwicklung der Plattform ließe es sich zudem implementieren, einzelne Programmbausteine mehrfach zu verwenden.

Aus meiner persönlichen Erfahrung heraus ist die Bedienung des RobotFramework Editors sehr umständlich. Prinzipiell lassen sich im Code-Editor Automatisierungen mit weniger Klicks implementieren als im BPMN-Editor, jedoch fehlen vor allem eine Auto-Vervollständigung sowie eine Schnellvorschau beim Schreiben des Codes. Aktuell existiert keine Möglichkeit, die Aktivitäten einer Library sowie die benötigten Parameter angezeigt zu bekommen. Daher ist das Programmieren momentan nur für Experten, die alle zu verwendenden Befehle kennen, möglich. In Zukunft kann die Weiterentwicklung des Robot Framework Language Servers¹⁵ eingebunden werden, die eine Code-Vervollständigung sowie Syntax-Validierung unterstützt.

Aus meiner Sicht bietet es sich an, für erfahrene Entwickler, die keine Erfahrung in der Modellierung haben, ein Low-Code-Interface wie Google Blockly¹⁶ bereitzustellen. Google Blockly ermöglicht die Programmierung in einem grafischen Code-Editor durch die Aneinanderreihung von Funktionsblöcken. Blockly ist unter anderem die dem grafischen Code-Editor zugrunde liegende Library des MIT AppInventors.

¹⁵<https://marketplace.visualstudio.com/items?itemName=robocorp.robotframework-lsp>

(Aufgerufen am 24.07.2021)

¹⁶<https://developers.google.com/blockly> (Abgerufen am 12.07.2021)

7. Zusammenfassung und Ausblick

In den eingangs dokumentierten Interviews wurde festgestellt, dass aktuell die Erstellung und Dokumentation von RPA-Robotern in verschiedenen Programmen erfolgen muss. Mit der vorgestellten Lösung ist es möglich, RPA-Roboter direkt in BPMN zu modellieren und die Automatisierungsfolgen zu implementieren. Zudem ist es durch die SSoT erstmals möglich, denselben Roboter in verschiedenen Modellierungsnotationen und Code-Editoren zu erstellen. Die einheitliche Speicherlösung und das Konzept der Parser ermöglichen darüber hinaus, weitere Modellierungsinterfaces, aber auch Code-Editoren zu ergänzen. Die Anbindung von Modellierungsinterfaces sowie zukünftig einem Low-Code-Editor, wie beispielsweise Google Blockly, ermöglicht auch für Prozessexperten ohne IT-Hintergrund die Erstellung von RPA-Robotern. Ebenso kann ein Entwickler den Roboter direkt im Code erstellen und trotzdem die für Prozessexperten relevante Dokumentation in Form eines BPMN-Diagramms bereitstellen. Sofern gewünscht, könnten Prozessexperten kleinere Änderungen an der Automatisierung in einem für sie geeigneteren Interface direkt vornehmen.

RPA-Automatisierungen sollten aus meiner Sicht in jedem Fall von RPA-Consultants begleitet werden. Andernfalls wird der Schritt der Prozessoptimierung (zum Beispiel mit Process Mining) oftmals vernachlässigt. Auch in den Interviews mit den RPA-Entwicklern wurde deutlich, dass vor jeder Automatisierung mögliches Optimierungspotenzial analysiert wird.

Die aktuelle Implementierung, die für jede Aktivität einen einzelnen Test-Case definiert, unterstützt keine lokalen Variablen, Verzweigungen und Schleifen. Um diese Konstrukte abbilden zu können, muss der RobotFramework-Code auf die Keywords-Syntax umgestellt werden. Wie der Code für unser Beispieldiagramm in dieser Syntax strukturiert ist, zeigt der Code-Ausschnitt in Abbildung 13. Die dort vorgestellte Syntax ermöglicht langfristig die Umsetzung aller in dieser Arbeit vorgestellten Konzepte. Durch die Trennung der RPA-Funktionen und der Programmstruktur ist der Code zudem besser lesbar. Sofern eine Umstellung auf die Keywords-Syntax erfolgt, ist zu prüfen, ob alle verwendeten Algorithmen zur Rekonstruktion des Graphenmodells korrekt funktionieren. Die bestehenden Tests decken nur sequenzielle Folgen von Anweisungen ab.

Eine durch die SSoT entstehende Einschränkung ist die Abstraktion der Modelle. So kann BPMN in der Standarddefinition deutlich mehr abbilden, als sich aufgrund des SSoT-Modells speichern lässt. Viele Details lassen sich nicht in den Code überführen oder in anderen Modellierungssprachen darstellen. Alle Eigenschaften, die sich nicht auf andere Modellierungsnotationen übertragen lassen, werden nicht in der SSoT gespeichert und stehen daher auch nicht in BPMN zur Verfügung. Dazu zählen unter anderem wesentliche Dokumentationsfeatures, aber auch Konzepte wie beispielsweise Pools, Lanes und aufklappbare Subprozesse. Letztere können, wie in Abbildung 1 erkenntlich, die Übersichtlichkeit der Prozessdarstellung erheblich verbessern. Zudem lassen sich über diese Syntax wiederverwendbare, modulare Roboter abbilden. Diese Funktion passt zur Anforderung von UiPath [AV], eine Integrationsmöglichkeit für Roboter aus der „globalen Anwender-Community“ anzubieten.

Ein möglicher Nachteil der SSoT-Lösung und der damit einhergehenden Nicht-Speicherung von interfacespezifischen Daten ist die automatische Ausrichtung aller Diagrammelemente bei jeder Öffnung des Modelers. Alle Elemente der visuellen Repräsentation - in BPMN - werden jedes Mal neu positioniert. Das kann einerseits gewollt sein, da zum Beispiel das `bpmn-js` CLI versucht, alle Elemente strukturiert anzuordnen. Andererseits wird keine manuell gewünschte Änderung in der Anordnung der Elemente gespeichert, wie sie beispielsweise zur besseren Übersicht beim Drucken des Diagramms gewünscht sein kann. Dieses Feature kann sich nur dann umgesetzt werden, wenn die in der Abbildung 6 dargestellte Architektur verwendet wird. Dann lässt sich jeder Roboter nur in genau einem Modellierungsinterface erstellen und zu jeder Ausführung in den RobotFramework Code parsen. Da ein Entwicklerteam mit genau einer Modellierungssprache arbeitet, kann sich diese Einschränkung auszahlen.

Abschließend ist festzuhalten, dass der Wunsch der RPA-Developer nach einer Plattform, die das Erstellen von Robotern mit verschiedenen Modellierungssprachen ermöglicht, erfüllt werden konnte. Hierzu wurde eingehend untersucht, welche Modellierungssprachen sich für die Erstellung der Roboter eignen. Durch die Proof of Concept Implementierung wurde das Konzept der einheitlichen Speicherlösung validiert. Mit der entstandenen Implementierung wurde eine Open-Source-RPA-Plattform geschaffen, die sich um weitere benötigte Interfaces erweitern lässt.

Literatur

- [Aal] AALST, Wil van d.: *Hybrid Intelligence: To Automate or Not to Automate, That is the question! (PEX Process Mining 2021)*. <https://youtu.be/ECPEJHN4YbY?t=3347>
- [Aal21] In: AALST, Wil van d.: *12 Process mining and RPA*. De Gruyter Oldenbourg, 2021, S. 223–240
- [AHH04] AALST, Wil van d. ; HEE, K.M. van ; HEE, K. van: *Workflow Management: Models, Methods, and Systems*. MIT Press, 2004 (Cooperative information systems). – ISBN 9780262720465
- [AR17] AGUIRRE, Santiago ; RODRIGUEZ, Alejandro": Automation of a Business Process Using Robotic Process Automation (RPA): A Case Study. In: *Applied Computer Sciences in Engineering*, Springer International Publishing, 2017, S. 65–71
- [AV] AP-VERLAG: *Diese sieben Kernanforderungen sollte eine RPA-Plattform mitbringen.* <https://ap-verlag.de/diese-sieben-kernanforderungen-sollte-eine-rpa-plattform-mitbringen/48518/>
- [FLL19] FLECHSIG, Christian ; LOHMER, Jacob ; LASCH, Rainer: Realizing the Full Potential of Robotic Process Automation Through a Combination with BPM. In: *Logistics Management*, Springer International Publishing, 2019, S. 104–119
- [FOSS18] In: FLEISCHMANN, Albert ; OPPL, Stefan ; SCHMIDT, Werner ; STARY, Christian: *Modellierungssprachen*. Wiesbaden : Springer Fachmedien Wiesbaden, 2018. – ISBN 978–3–658–22648–0, 71–128
- [Gar] GARTNER: *2020 Gartner Magic Quadrant for Robotic Process Automation*. <https://www.uipath.com/de/company/rpa-analyst-reports/gartner-magic-quadrant-robotic-process-automation>
- [Her84] In: HERING, Ekbert: *Programmablaufplan nach DIN 66001*. Wiesbaden : Vieweg+Teubner Verlag, 1984. – ISBN 978–3–322–86222–8, 26–34

- [ISG] ISG: *Robotic Process Automation (RPA) sorgt für mehr Produktivität und nicht für Jobverluste.* https://isg-one.com/docs/default-source/default-document-library/isg-automation-index-de_final_form.pdf?sfvrsn=15defe31_0
- [Lob] LOBE, Christopher: *Bewertung der Eignung von Modellierungssprachen zur ebenenübergreifenden Prozessdarstellung im Managementhandbuch*, Universität Magdeburg, Diplomarbeit
- [MCKL11] MINONNE, Clemente ; COLICCHIO, Carlo ; KELLER, Thomas ; LITZKE, Matthias: *Business Process Management 2011 - Status quo und Zukunft : Eine empirische Studie im deutschsprachigen Europa*. Version: 01 2011. <http://dx.doi.org/10.21256/zhaw-1026>. – DOI 10.21256/zhaw-1026
- [NR02] NÜTTGENS, Markus ; RUMP, Frank J.: Syntax und semantik ereignisgesteuerter prozessketten (epk). In: *Promise 2002–Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen* (2002)
- [PwC] PwC: *Robotic Process Automation (RPA) in der DACH-Region.* <https://www.pwc.de/de/rechnungslegung/robotic-process-automation-rpa-in-der-dach-region.pdf>
- [Via] VIADEE: *Robotic Process Automation mit BPMN und CAMUNDA.* <https://blog.viadee.de/robotic-process-automation-mit-bpmn-und-camunda>
- [Wil07] WILHELM, Rudolf: *Prozessorganisation*. Oldenbourg Verlag, 2007
- [Win00] WINTER, Andreas: *Referenz-Metaschema für visuelle Modellierungssprachen*. Dt. Univ.-Verlag, 2000

A. Anhang

```
*** Settings ***
Library           RPA.Excel.Files
Library           RPA.Outlook.Application
Library           RPA.JSON

*** Tasks ***
Open Excel
  Open Workbook    ${CURDIR}${/}reminder.xlsx
Open Outlook
  Open Application
Read Worksheet
  ${worksheet} =  Read Worksheet As Table    header=true
FOR  ${row}  IN  @{worksheet}
  LOG    ${row}
  ${lastname} =  Get value from JSON    ${row}    $.Lastname
  ${firstname} =  Get value from JSON    ${row}    $.Firstname
  ${mail} =  Get value from JSON    ${row}    $.Mail
  ${counter} =  Get value from JSON    ${row}    $.Counter
  IF    ${counter} == 1
    Send Message    ${mail}    First Reminder    Dear ${lastname}, ...
                    False    ${CURDIR}${/}1.pdf
  ELSE IF   ${counter} == 2
    Send Message    ${mail}    Second Reminder    Dear ${lastname}, ...
                    False    ${CURDIR}${/}2.pdf
  END
END
Close Outlook
  Quit Application
Close Workbook
  Close Workbook
```

Abbildung 12: Beispielprozess in RobotFramework-Syntax

```

*** Settings ***
Library          RPA.Excel.Files
Library          RPA.JSON
Library          RPA.Outlook.Application

*** Tasks ***
RPA-Bot
    Open Excel
    Open Outlook
    ${worksheet} =  Get Data from Excel
    FOR ${row} IN @${worksheet}
        Log Row  ${row}
        ${lastname} =  Get Lastname  ${row}
        ${firstname} =  Get Fristname  ${row}
        ${mail} =  Get Mail  ${row}
        ${counter} =  Get Counter  ${row}
        IF  ${counter} == 1
            Send Reminder1  ${mail}  ${lastname}
        ELSE IF  ${counter} == 2
            Send Reminder2  ${mail}  ${lastname}
        END
    END
    Close Outlook
    Close Excel

*** Keywords ***
Open Outlook
    Open Application
Open Excel
    Open Workbook  ${CURDIR} ${}/reminder.xlsx
Send Reminder1
    [Arguments]  ${mail}  ${lastname}
    Send Message  ${mail}  First Reminder  Dear ${lastname}, ...  False  ${CURDIR} ${}/1.pdf
Send Reminder2
    [Arguments]  ${mail}  ${lastname}
    Send Message  ${mail}  Second Reminder  Dear ${lastname}, ...  False  ${CURDIR} ${}/2.pdf
Get Data from Excel
    ${worksheet} =  Read Worksheet As Table  header=true
    [return]  ${worksheet}
Log Row
    [Arguments]  ${row}
    LOG  ${row}
Get Lastname
    [Arguments]  ${row}
    ${lastname} =  Get values from JSON  ${row}  $.Lastname
    [return]  ${lastname}
Get Fristname
    [Arguments]  ${row}
    ${firstname} =  Get values from JSON  ${row}  $.Firstrname
    [return]  ${firstname}
Get Mail
    [Arguments]  ${row}
    ${mail} =  Get value from JSON  ${row}  $.Mail
    [return]  ${mail}
Get Counter
    [Arguments]  ${row}
    ${counter} =  Get value from JSON  ${row}  $.Counter
    [return]  ${counter}
Close Outlook
    Quit Application
Close Excel
    Close Workbook

```

Abbildung 13: Beispielprozess in RobotFramework-Syntax unter Verwendung der
Keywords

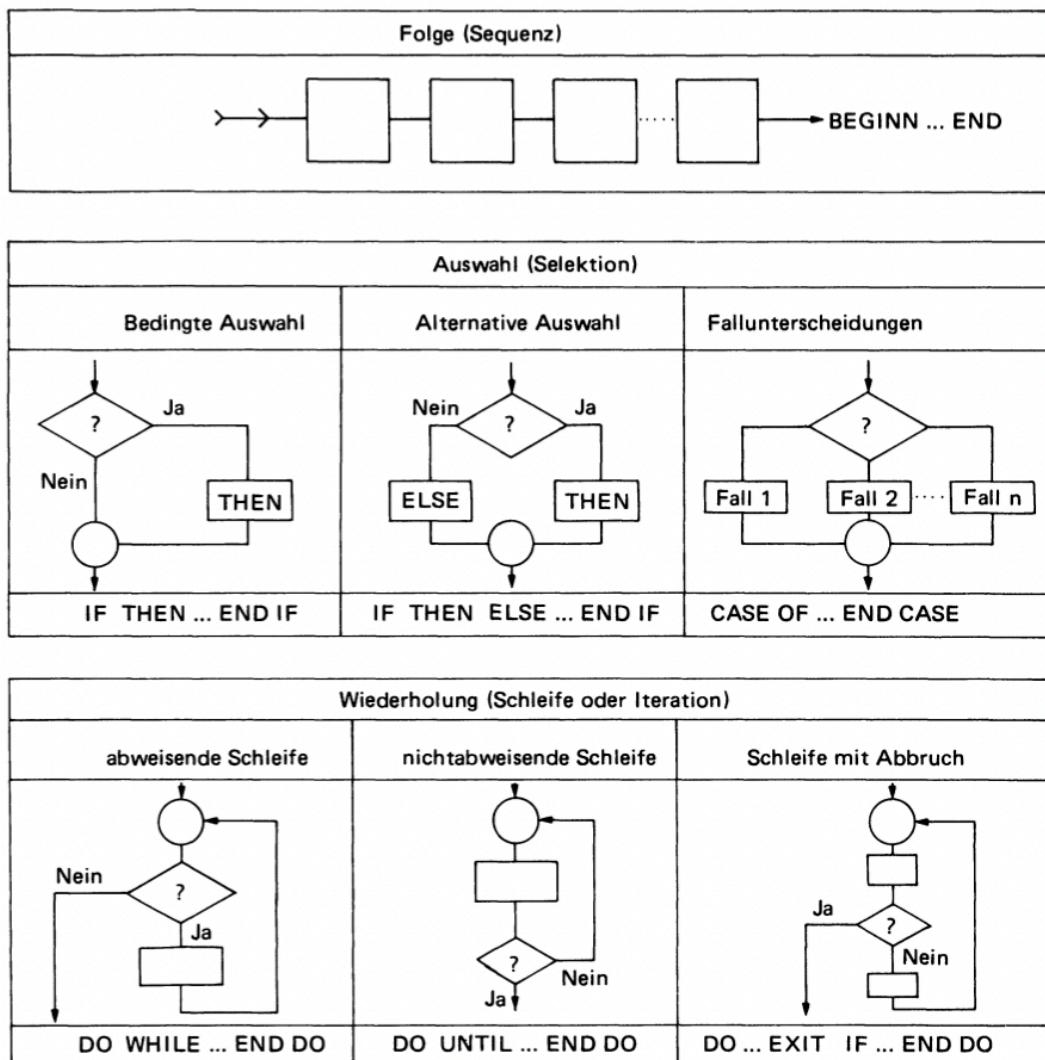


Abbildung 14: Auflistung der elementaren logischen Ablaufstrukturen