profi.com®
we make IT work

SUCCESSSTORY

ERSTELLUNG EINER SKALIERBAREN TESTAUTOMATISIERUNGS-LÖSUNG BEI EDEKA MINDEN-HANNOVER

Im April 2017 wurde die profi.com AG durch die EDEKA Minden-Hannover IT-Service GmbH für einen Proof of Concept (PoC) zur Testautomatisierung von technologieübergreifenden En-to-End-Testfällen beauftragt. Im Folgenden wollen wir die bisherigen Ergebnisse dieses Projekts vorstellen. Des Weiteren soll aufgezeigt werden, wie durch einen strukturierten Framework-Aufbau die Regionalgesellschaft EDEKA Minden-Hannover in die Lage versetzt wurde, jederzeit geschäftskritische Prozesse qualitätssichernd bewerten zu können.

Der PoC umfasste die Pilotierung der Automatisierung mittels Micro Focus Unified Functional Testing (ehem. HPE UFT) sowie die Erstellung eines ersten Grobkonzepts zur finalen Einführung.

Nachdem einige Konzepte und Tools gegenübergestellt wurden, entschied sich die EDEKA Minden-Hannoverzur Nutzung von UFT als Hauptwerkzeug. Vor allem die Möglichkeit zur Automatisierung anwendungs- und technologie-übergreifender Prozesse war bei der Entscheidungsfindung von grundlegender Bedeutung.

Im zweiten Teil dieser Success Story zeigen wir eine Lösung auf, wie durch den Einsatz eines asynchronen Verteilverfahrens die Testausführung skalierbar eingesetzt werden kann, um Laufzeiten und Feedback-Zyklen zu minimieren.

Die EDEKA Minden-Hannover ist mit einem Umsatz von 9,36 Milliarden Euro und mehr als 72.200 Mitarbeiterinnen und Mitarbeitern einschließlich des selbstständigen Einzelhandels die umsatzstärkste Regionalgesellschaft im genossenschaftlich organisierten EDEKA-Verbund. Das Geschäftsgebiet erstreckt sich von der niederländischen bis an die polnische Grenze, es umfasst einen Teil von Ostwestfalen-Lippe, nahezu vollständig Niedersachsen, Bremen, Sachsen-Anhalt, Berlin und Brandenburg. Zwei Drittel der mehr als 1.500 Märkte sind in der Hand von selbstständigen EDEKA-Einzelhändlern. Außerdem gehören fünf Produktionsbetriebe für Brot- und Backwaren (Schäfer's) sowie zwei Produktionsbetriebe für Fleisch- und Wurstwaren (Bauerngut) und ein Produktionsbetrieb für Frischfisch und Fischprodukte (Hagenah) zur EDEKA Minden-Hannover.

Die EDEKA Minden eG wurde 1920 als genossenschaftlicher Verbund von selbstständigen Kaufleuten gegründet. Heute zählen rund 580 Genossenschaftsmitglieder zur EDEKA Minden eG.

Die EDEKA Minden-Hannover IT-Service GmbH verantwortet als Tochtergesellschaft die Informationstechnologie des Unternehmensverbundes und betreibt als Dienstleistungsgesellschaft alle IT-Systeme für den Groß- und Einzelhandel sowie für die Produktionsbetriebe.

Weitere öffentliche Informationen können über https://verbund.edeka/minden-hannover/ gefunden werden.

Micro Focus UFT als Automat für Cross-Application Automation

Wie bereits angedeutet, bestand der Auftrag nicht in der vergleichsweise weniger komplexen Automatisierung von anwendungsspezifischen Testfällen. Stattdessen wurden über die letzten Monate diverse technologie-übergreifende (Cross-Application) End-to-End-Prozesse automatisiert.

Durch ein fehlendes Micro Focus Application Lifecycle Management (ALM), war die Erstellung eines skalierbaren und wartungsarmen Frameworks auf Basis von Business Process Testing (BPT) nicht möglich.

Alternativ wählten wir einen Ansatz, welcher sich ausschließlich auf Funktionalitäten innerhalb UFTs stützt. Wiederverwendbare Funktionen wurden durch Run Action-Aufrufe der darüber liegenden Schichten realisiert. Innerhalb des Schicht-Modells wurde eine harte Trennung zwischen Fachlichkeit und Technik implementiert.

Visualisierung eines Schicht-Modells

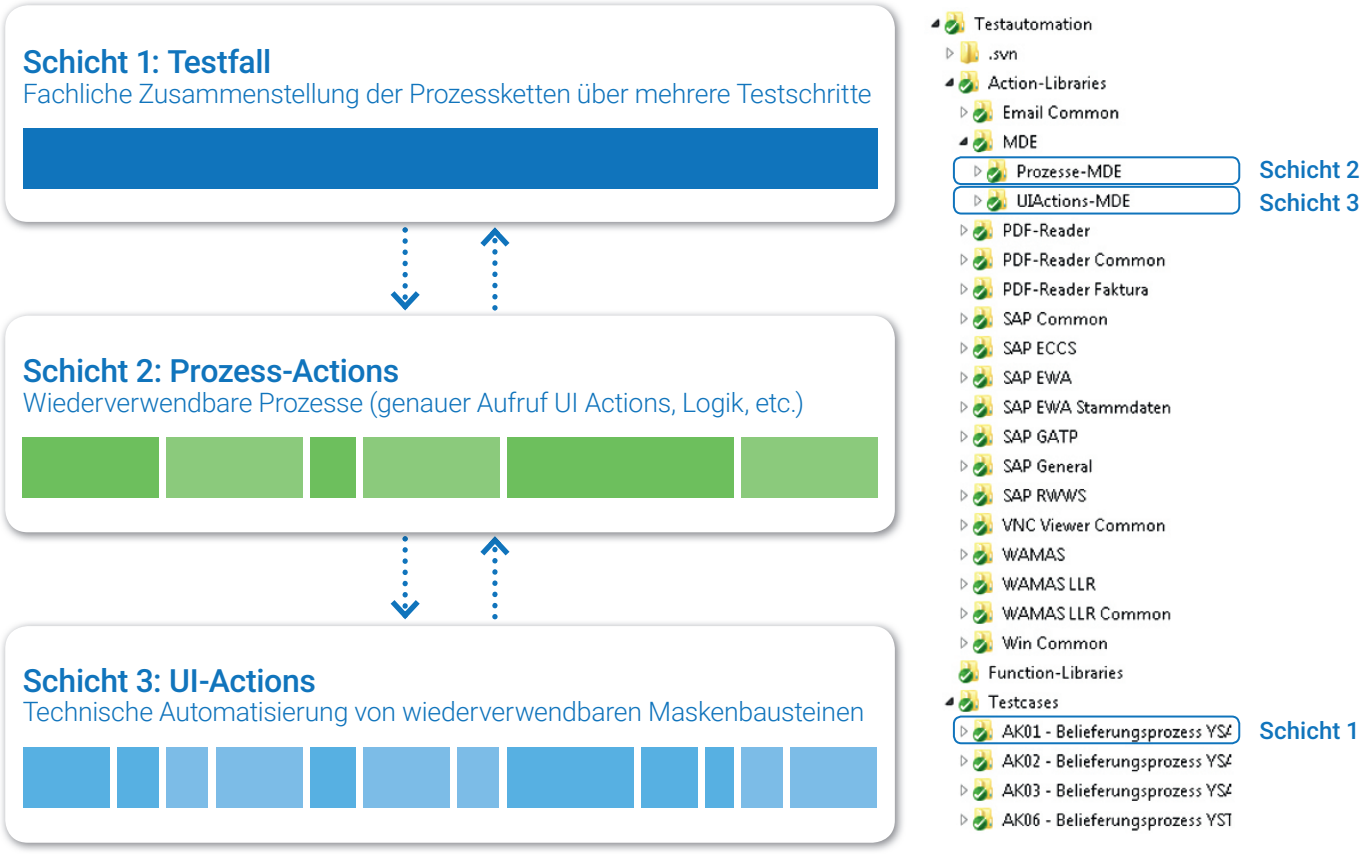


Abb. 1: Visualisierung des Schicht-Modells und Implementierung im Dateisystem

- Die **Testfallschicht (Schicht 1)** definiert lediglich die Reihenfolge und Datenübergabe benötigter (Sub-)Prozesse und stößt entsprechende Actions an.
- Innerhalb der **Prozessschicht (Schicht 2)** werden diese Prozesse genauer definiert.
- Die entsprechenden Aktionen auf den System-Oberflächen werden durch Aufrufe der **UI-Actions-Schicht (Schicht 3)** durchgeführt. Erst hier werden wirkliche Systemtrennungen in den Skripten definiert.

Gründe für ein Schicht-Modell

Die Grundidee des Konzepts wurde aus BPT entnommen: Die Trennung zwischen fachlicher Definition des Testfalles und technischem Durchgriff auf die Oberflächen folgt damit üblichen Best Practices

vgl. https://www.proficom.de/unternehmen/aktuelles/artikel/top-7-best-practices-in-der-testautomatisierung.html

Innerhalb eines derartigen Schicht-Modells ist es möglich, Änderungen an den Oberflächen zentral in so genannten UI-Actions zu ändern. Neben dem Code für den Zugriff auf Elemente, befinden sich auch die Objekt-Repositories in den entsprechenden Aktionen.

Dadurch war es uns in Schicht 3 möglich eine Trennung der Skripte für insgesamt zehn Systeme innerhalb der Prozesse zu etablieren. Neben sieben SAP-Systemen besteht die zu testende Infrastruktur aus den folgenden Applikationen:

- System für »Mobile Datenerfassung (MDE)«
  (VNC-Viewer, realisiert über Visual Identification)
- WAMAS® der Firma SSI Schäfer in der Version 4
  (Leitstandlösung und Emulation des Funklclients, beide .NET/C++)
- PDF Reader

Fachliche Anpassungen (zusätzliche Prüfungen, Änderungen des Prozesses etc.) lassen sich dagegen einfach in den Schichten 1 und 2 implementieren. Eine Anpassung der techniknahen UI-Actions ist nicht notwendig.

Herausforderungen

Bei jeder hinreichend komplexen Architektur gibt es Herausforderungen, die gelöst werden müssen.

Trotz einer (theoretischen) Trennung zwischen fachlichem und technischem Code müssen alle Skripte als UFT-Testfälle geschrieben werden. Eine wirkliche scriptless-Automation, wie sie durch BPT versprochen wird, ist dementsprechend leider nicht möglich. Allerdings gibt dieser Umstand auch wieder größere Flexibilität auf den höheren Framework-Ebenen. Zum Beispiel müssen für Daten-Transformationen (z.B. Auslesen eines Sub-Strings, Umformatierung eines Datums etc.) beim Einsatz von BPT eigene Komponenten entwickelt werden. Im Framework der EDEKA Minden-Hannover können derartige Schritte bei Bedarf in das Testfall-Skript eingefügt werden.

Mit der Zeit zeichnete sich eine weitere Herausforderung ab: Ladezeiten. Durch Multiplikator-Effekte beim Aufruf von Action-Calls innerhalb des Testfalles kam es vor, dass einige Testfälle bis zu eine Minute Ladezeiten im Editor aufzeigten. Das Verhalten konnte durch den Einsatz des LoadAndRunAction-Aufrufs, welcher abhängige Aktionen erst zur Laufzeit einliest und öffnet, abgeschwächt werden.

Mit Stand Februar 2019 wurden 20 Tests mit einer durchschnittlichen Laufzeit von ca. 30 Minuten pro Testfall realisiert, was zu einer Gesamtlaufzeit von rund 10 Stunden führt. Die vergleichsweise lange Laufzeit eines Testfalles ist dem Umfang und der Komplexität eines End-to-End-Prozesses geschuldet.

Skalierung der Laufzeiten

Bei 10 Stunden Laufzeit auf einem Host wurde schnell klar, dass ein wirklicher Mehrwert nur über Parallelisierung zu erreichen ist. Dies beinhaltete vor allem folgende Anforderungen:

- Ausführungen müssen parallel auf verschiedenen, dezentralen Hosts gestartet werden
- Ergebnisse müssen nach Abschluss aller Tests zentral aggregiert und reportet werden
- Code muss auf allen Hosts aktuell gehalten werden

Die dann später umgesetzte Lösung sah ein Server-Agenten-System vor, welches durch eine zentrale Stelle (Server) Testaufträge in einem gemeinsam genutzten Netzlaufwerk ablegt. Die einzelnen Clients (Agenten) prüfen asynchron, ob Testaufträge auf dem Netzlaufwerk vorliegen. Sollte ein neuer Auftrag vorliegen, wird der darin definierte Testfall durchgeführt.

1 Anlegen von Jobs

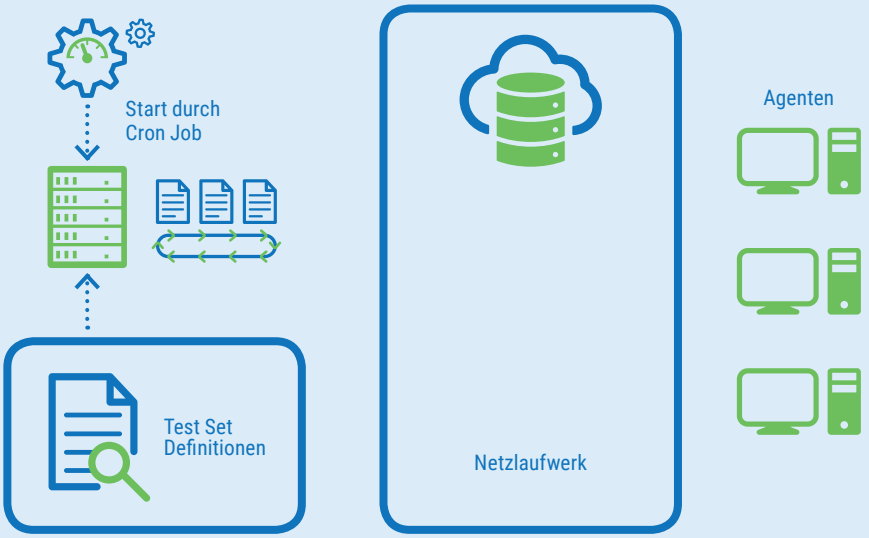
Gestartet durch einen Cronjob öffnet der Server eine per Konfiguration festgelegte Liste an Szenarien. Die darin enthaltenden Tests werden als Job auf einem Netzlaufwerk abgelegt. Ein Szenario beinhaltet eine Liste der Testfälle mit folgenden Informationen:

- Name des Testfalls
- Bevorzugter Agent (für voneinander abhängige Testfälle)

Jeder Eintrag dieser Liste wird in ein Job-File umgewandelt und auf dem Netzlaufwerk abgelegt. Ein initiales Job-File beinhaltet folgende Informationen zur Auswertung an den Agenten:

- Name des Testfalls

Start der Testausführung

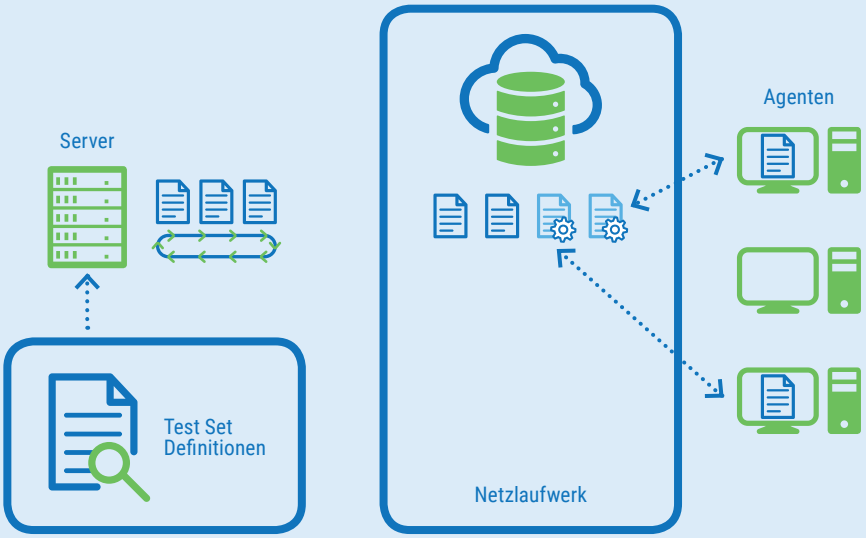


2 Aufnehmen und Durchführen von Jobs

Jeder Agent verbindet sich in regelmäßigen Abständen mit dem Netzlaufwerk und prüft, ob Testaufträge vorliegen. Dabei wird zuerst ein Agenten-spezifischer Ordner geprüft, danach ein gemeinsamer Auftragsordner.

Sollten freie Aufträge vorliegen, übernimmt ein Agent den nächstbesten Job und markiert diesen als »in Bearbeitung«.

Agenten starten die Abarbeitung

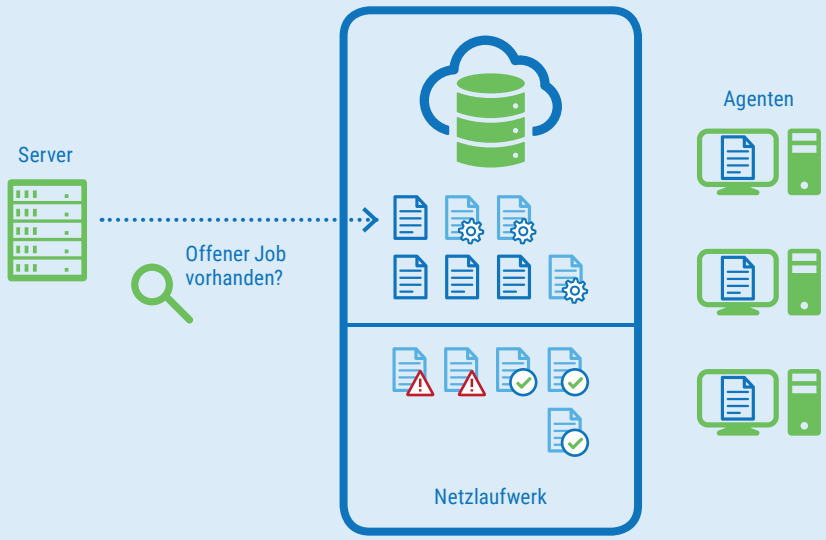


Nach der Abarbeitung eines Jobs reichern die Agenten das Job-File mit den nachfolgenden Informationen an und markieren es als erledigt:

- Name des Testfalls
- Ausführender Agent
- Zeitstempel für Start und Ende der Testfalldurchführung
- Testfall-Status (passed/failed)
- Netzwerkpfad des Reports
- Meldung im technischen Fehlerfall (z.B. Testfall konnte nicht geöffnet werden)

Ein Großteil dieser Daten wird durch UFT bereits geliefert und muss lediglich in die Datei übertragen werden. Anschließend wird der nächste freie Job durch den Agenten gestartet. Parallel zur Ausführung prüft der Server regelmäßig, ob alle Aufträge bearbeitet wurden, um anschließend in die Testende-Phase überzugehen.

Server prüft auf Abschluss der Arbeiten



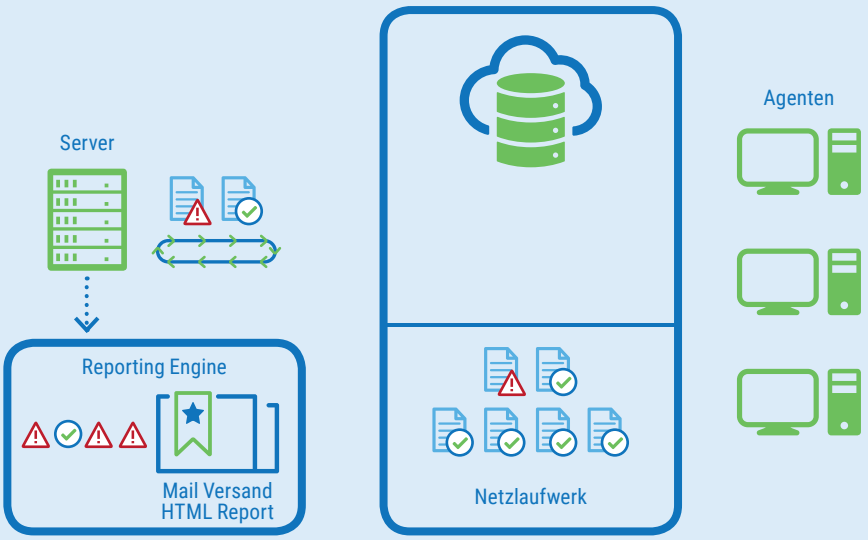
3 Testende und Reporting

Sobald kein Auftrag mehr offen oder in Bearbeitung ist, werden alle Job-Files geprüft und die Ergebnisse an das Reporting übergeben. Die Reporting Engine transformiert die vorliegenden Informationen:

- Anzahl durchgeführter Testfälle
- Anzahl erfolgreicher und fehlgeschlagener Testfälle
- Laufzeiten
- Detail-Ergebnisse

in eine aggregierte E-Mail, welche an einen vorher konfigurierten Verteiler versandt wird. Zusätzlich werden durch XML-Transformation die einzelnen Ergebnisse der Testfälle als HTML-Report zusammengefasst und auf einem zentral verfügbaren Netzlaufwerk zur Verfügung gestellt.

Testende und Report-Erstellung

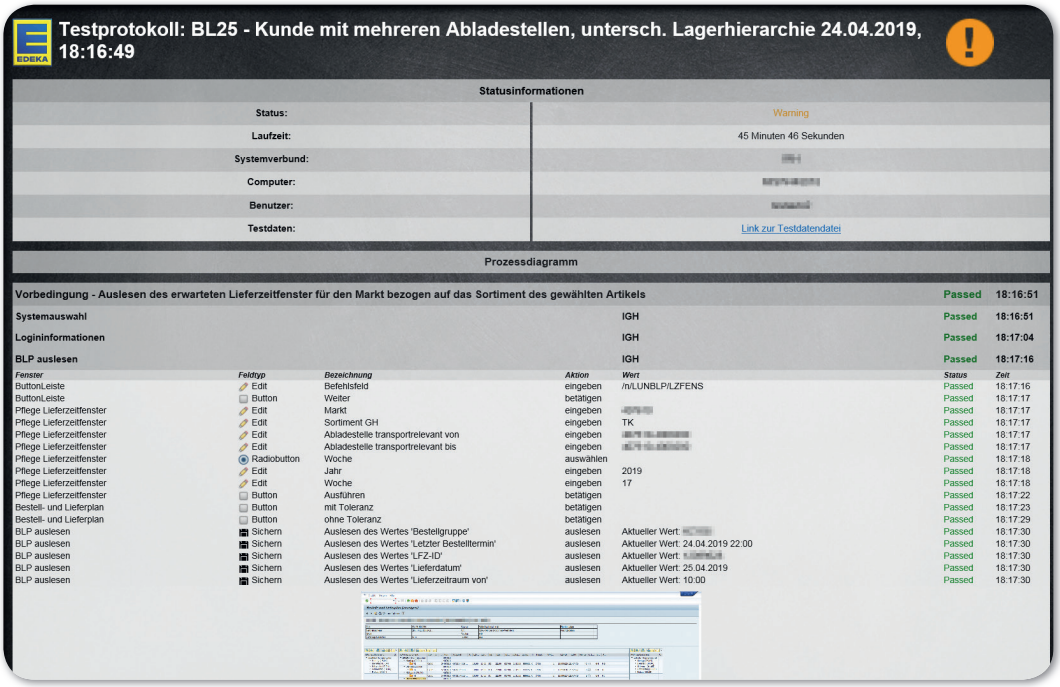


Fazit und Ausblick

Neben der erfolgreichen Einführung einer komplexen End-to-End-Testautomatisierung ist es uns mit dem **Einsatz eines selbst erstellten Scheduler- und Verteil-Mechanismus** gelungen, die Laufzeit der Testfälle signifikant zu senken.

Der derzeitige Einsatz der Lösung sieht einen **Verbund von fünf parallelen Sessions (Clients) auf einem Windows-Server** vor, was die **Laufzeiten der Testfälle auf ungefähr 20 bis 25 % der ursprünglichen Zeiten senken** wird. Mittelfristig ist ein Ausbau der Testautomatisierung mit weiteren End-to-End-Prozessen vorgesehen. In einer nächsten Ausbaustufe soll der Testfallbestand um weitere granulare Anwendungs-Testfälle erweitert werden.

Die Testfälle werden in einem agilen Entwicklungsprojekt nächtlich durchgeführt, um die täglichen Lieferungen in die Testumgebung zu überprüfen. Zu Sprint-Ende soll durch untertägige Tests eine Aussage zur Auslieferbarkeit des Releases gegeben werden.



Testprotokoll: BL25 - Kunde mit mehreren Abladestellen, untersch. Lagerhierarchie 24.04.2019, 18:16:49

Statusinformationen

Status:	Warning
Laufzeit:	45 Minuten 46 Sekunden
Systemverbund:	
Computer:	
Benutzer:	
Testdaten:	Link zur Testdatendatei

Prozessdiagramm

Vorbedingung - Auslesen des erwarteten Lieferzeitfenster für den Markt bezogen auf das Sortiment des gewählten Artikels					Passed	18:16:51
Systemauswahl				IGH	Passed	18:16:51
Logininformationen				IGH	Passed	18:17:04
BLP auslesen				IGH	Passed	18:17:16
Fenster	*Feldtyp*	*Bezeichnung*	*Aktion*	*Wert*	*Status*	*Zeit*
ButtonLeiste	Edit	Befehlsfeld	eingeben	/n/LUNBLP/LZFENS	Passed	18:17:16
ButtonLeiste	Button	Weiter	betätigen		Passed	18:17:17
Pflege Lieferzeitfenster	Edit	Markt	eingeben		Passed	18:17:17
Pflege Lieferzeitfenster	Edit	Sortiment GH	eingeben	TK	Passed	18:17:17
Pflege Lieferzeitfenster	Edit	Abladestelle transportrelevant von	eingeben		Passed	18:17:17
Pflege Lieferzeitfenster	Edit	Abladestelle transportrelevant bis	eingeben		Passed	18:17:17
Pflege Lieferzeitfenster	Radiobutton	Woche	auswählen		Passed	18:17:18
Pflege Lieferzeitfenster	Edit	Jahr	eingeben	2019	Passed	18:17:18
Pflege Lieferzeitfenster	Edit	Woche	eingeben	17	Passed	18:17:18
Pflege Lieferzeitfenster	Button	Ausführen	betätigen		Passed	18:17:22
Bestell- und Lieferplan	Button	mit Toleranz	betätigen		Passed	18:17:23
Bestell- und Lieferplan	Button	ohne Toleranz	betätigen		Passed	18:17:29
BLP auslesen	Sichern	Auslesen des Wertes 'Bestellgruppe'	auslesen	Aktueller Wert:	Passed	18:17:30
BLP auslesen	Sichern	Auslesen des Wertes 'Letzter Bestelltermin'	auslesen	Aktueller Wert: 24.04.2019 22:00	Passed	18:17:30
BLP auslesen	Sichern	Auslesen des Wertes 'LFZ-ID'	auslesen	Aktueller Wert:	Passed	18:17:30
BLP auslesen	Sichern	Auslesen des Wertes 'Lieferdatum'	auslesen	Aktueller Wert: 25.04.2019	Passed	18:17:30
BLP auslesen	Sichern	Auslesen des Wertes 'Lieferzeitraum von'	auslesen	Aktueller Wert: 10:00	Passed	18:17:30

Abb. 6: Detail-Protokoll einer Testfallausführung

Zusammenfassung:

Datum der Testdurchführung:	24.04.2019
Start der Testdurchführung:	17:05 Uhr
Ende der Testdurchführung:	19:44 Uhr
Gesamtlaufzeit:	02:39 h
Umfang des TestSets:	15 Test(s)
Erfolgreich geladene Tests:	15 Test(s)
Nicht erfolgreich geladene Tests:	0 Test(s)
Testfälle Status Passed:	10 Test(s)
Testfälle Status Warning:	1 Test(s)
Testfälle Status Failed:	4 Test(s)

Detailergebnisse:

Nr.	Testfall	Status	Startzeit	Laufzeit	Protokoll
1	AK98 - Automatische Aktionsanlage	Passed	17:05:18	00:07:34	Link
2	BL99 - Löschen von Dummy-Aufträgen	Passed	17:05:29	00:01:05	Link
3	AK99 - Automatische Aktionsdeaktivierung	Passed	17:05:41	00:00:45	Link
4	AK05 - Belieferungsprozess YSTA mit Aktionsnummer inkl. Null- und Mindermengen und Variation Regal- und Ordersatz	Passed	17:06:36	00:57:22	Link
5	AK06 - Belieferungsprozess YSTA mit Aktionsnummer Vollbestätigung	Failed	17:07:40	00:57:46	Link
6	BL01 - Belieferungsprozess Vollbestätigung über MDE	Passed	17:08:21	00:55:23	Link
7	BL04 - Kundenauftrag Teil-Nullmenge LVS	Failed	17:08:53	01:06:43	Link
8	AK02 - Belieferungsprozess YSA1 Vollbestätigung inkl. Null-Mindermenge und Börsenrabatt	Passed	17:13:36	01:03:10	Link
9	BL08 - Geplanter Selbstabholer	Passed	18:05:00	00:38:50	Link
10	BL10 - VN ohne Lokationssubstitution	Passed	18:05:38	00:59:05	Link
11	BL14 - Ersatzartikel	Failed	18:06:12	00:58:01	Link
12	BL25 - Kunde mit mehreren Abladestellen, untersch. Lagerhierarchie	Warning	18:16:49	00:46:28	Link
13	AK31 - Ersatzartikel	Passed	18:17:55	00:48:32	Link
14	BL26 - Manueller Sofortauftrag YSMB über ZAB_SCHNELLERFASSUNG	Passed	18:45:07	00:32:36	Link
15	BL05 - Huckepacktour (geplant)	Failed	19:04:27	00:40:19	Link

Die Testergebnisse für den Tag finden sich an folgendem Ablageort:

Abb. 7: zusammenfassendes Protokoll eines nächtlichen Testlaufes per Mail

Der Ablauf und die Umsetzung des Projektes zeigten uns, dass iterative Vorgehen, auch bei großen Feedback-Zyklen, eine erfolgsversprechende und zielführende Herangehensweise darstellen. Später identifizierte Probleme, wie Laufzeiten, Ladezeiten der Testfälle und diverse Instabilitäten der Verlinkungen konnten durch die gemeinsame Zusammenarbeit der EDEKA Minden-Hannover und der profi.com AG analysiert, stabilisiert und behoben werden.

Wir sind stolz und glücklich, die Möglichkeit für dieses Projekt durch die EDEKA Minden-Hannover erhalten zu haben und bedanken uns an dieser Stelle recht herzlich für das in uns gesetzte Vertrauen.