



TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Fakultät Informatik

**Konzeption und Realisierung
eines Programms
zur Ermittlung von Objektabhängigkeiten
innerhalb eines
SAP-Software-Transportauftrags**

Bachelorarbeit im Studiengang Informatik

vorgelegt von

Felix Berger

Matrikelnummer: 3272014

Erstgutachter: Prof. Dr. Rainer Weber

Zweitgutachter: Prof. Dr. Joachim Scheja

Externer Betreuer: Alexander Kreutner (Dematic GmbH)

© 2022

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

Hinweis: Diese Erklärung ist in alle Exemplare der Abschlussarbeit fest einzubinden. (Keine Spiralbindung)

Prüfungsrechtliche Erklärung der/des Studierenden

Angaben des bzw. der Studierenden:

Name: Vorname: Matrikel-Nr.:

Fakultät: Studiengang:

Sommersemester Wintersemester

Titel der Abschlussarbeit:

Ich versichere, dass ich die Arbeit selbständig verfasst, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum, Unterschrift Studierende/Studierender

Erklärung zur Veröffentlichung der vorstehend bezeichneten Abschlussarbeit

Die Entscheidung über die vollständige oder auszugsweise Veröffentlichung der Abschlussarbeit liegt grundsätzlich erst einmal allein in der Zuständigkeit der/des studentischen Verfasserin/Verfassers. Nach dem Urheberrechtsgesetz (UrhG) erwirbt die Verfasserin/der Verfasser einer Abschlussarbeit mit Anfertigung ihrer/seiner Arbeit das alleinige Urheberrecht und grundsätzlich auch die hieraus resultierenden Nutzungsrechte wie z.B. Erstveröffentlichung (§ 12 UrhG), Verbreitung (§ 17 UrhG), Vervielfältigung (§ 16 UrhG), Online-Nutzung usw., also alle Rechte, die die nicht-kommerzielle oder kommerzielle Verwertung betreffen.

Die Hochschule und deren Beschäftigte werden Abschlussarbeiten oder Teile davon nicht ohne Zustimmung der/des studentischen Verfasserin/Verfassers veröffentlichen, insbesondere nicht öffentlich zugänglich in die Bibliothek der Hochschule einstellen.

Hiermit genehmige ich, wenn und soweit keine entgegenstehenden Vereinbarungen mit Dritten getroffen worden sind,
 genehmige ich nicht,

dass die oben genannte Abschlussarbeit durch die Technische Hochschule Nürnberg Georg Simon Ohm, ggf. nach Ablauf einer mittels eines auf der Abschlussarbeit aufgetragenen Sperrvermerks kenntlich gemachten Sperrfrist

von Jahren (0 - 5 Jahren ab Datum der Abgabe der Arbeit),

der Öffentlichkeit zugänglich gemacht wird. Im Falle der Genehmigung erfolgt diese unwiderruflich; hierzu wird der Abschlussarbeit ein Exemplar im digitalisierten PDF-Format auf einem Datenträger beigelegt. Bestimmungen der jeweils geltenden Studien- und Prüfungsordnung über Art und Umfang der im Rahmen der Arbeit abzugebenden Exemplare und Materialien werden hierdurch nicht berührt.

Ort, Datum, Unterschrift Studierende/Studierender

Inhaltsverzeichnis

| | | |
|-------|--|----|
| 1 | Einleitung | 1 |
| 1.1 | Motivation | 1 |
| 1.2 | Zielsetzung | 1 |
| 1.3 | Struktur und Aufbau | 2 |
| 2 | Theoretische Grundlagen | 3 |
| 2.1 | Projekt | 3 |
| 2.2 | ABAP | 3 |
| 2.3 | Dematic Project And Transport Management System | 6 |
| 2.4 | Paketprüfung | 7 |
| 3 | Anwendungsplanung | 8 |
| 3.1 | Anforderungsanalyse | 8 |
| 3.2 | Bestimmung des geeignetsten prototypischen Vorgehensmodells | 9 |
| 3.2.1 | Vorstellung der gängigsten prototypischen Vorgehensmodelle | 9 |
| 3.2.2 | Auswahl eines prototypischen Vorgehensmodells | 10 |
| 3.3 | Anwendungskonzeption | 11 |
| 3.3.1 | Konzept der Paketprüfung | 11 |
| 3.3.2 | Konzept der Benutzerschnittstelle | 14 |
| 3.3.3 | Modultestkonzept | 14 |
| 4 | Umsetzung | 15 |
| 4.1 | Prüfung von Domänen | 15 |
| 4.1.1 | Die Benutzerschnittstelle – Eingabemaske | 15 |
| 4.1.2 | Die Serviceklasse | 16 |
| 4.1.3 | Modultestergebnis | 17 |
| 4.1.4 | Übersicht über den Prüfprozess einer Domäne | 18 |
| 4.2 | Prüfung von Datenelementen | 19 |
| 4.2.1 | Modultestergebnis | 20 |
| 4.2.2 | Übersicht über den Prüfprozess eines Datenelements | 21 |
| 4.3 | Prüfung von Tabellentypen | 21 |
| 4.3.1 | Modultestergebnis | 22 |
| 4.3.2 | Übersicht über den Prüfprozess eines Tabellentyps | 23 |
| 4.4 | Prüfung von Tabellen und Strukturen | 24 |
| 4.4.1 | Modultestergebnis | 25 |
| 4.4.2 | Übersicht über den Prüfprozess einer Tabelle bzw. einer Struktur | 26 |
| 4.5 | Finale Anwendungsarchitektur | 27 |
| 4.5.1 | Die Benutzerschnittstelle – Protokollausgabe | 27 |

| | | |
|-------|---------------------------|----|
| 4.5.2 | Testergebnis | 29 |
| 5 | Schluss | 30 |
| 5.1 | Zusammenfassung | 30 |
| 5.2 | Fazit..... | 31 |
| 5.3 | Ausblick | 31 |
| | Literaturverzeichnis..... | 33 |

Darstellungsverzeichnis

| | |
|--|----|
| Darstellung 1: Erklärung der wesentlichsten, für diese Ausarbeitung relevanten Datenbanktabellen | 5 |
| Darstellung 2: Paketauswahl unter Zuhilfenahme des Dematic TB..... | 6 |
| Darstellung 3: Beispielhafter Aufbau einer tiefen Struktur..... | 12 |
| Darstellung 4: Beispielhafter Aufbau eines Tabellentyps | 14 |
| Darstellung 5: Konzept der Benutzerschnittstelle | 14 |
| Darstellung 6: Eingabemaske des Dynpros zur Abfrage der Nummer des Transportauftrags | 15 |
| Darstellung 7: Beispielhaftes Ergebnis einer Paketprüfung von Domänen | 17 |
| Darstellung 8: Flussdiagramm des Prüfprozesses einer Domäne..... | 18 |
| Darstellung 9: Beispielhaftes Ergebnis einer Paketprüfung von Datenelementen | 20 |
| Darstellung 10: Flussdiagramm des Prüfprozesses eines Datenelements..... | 21 |
| Darstellung 11: Beispielhaftes Ergebnis der Paketprüfung eines Tabellentyps..... | 22 |
| Darstellung 12: Flussdiagramm des Prüfprozesses eines Tabellentyps | 23 |
| Darstellung 13: Beispielhaftes Ergebnis der Paketprüfung einer Struktur | 25 |
| Darstellung 14: Flussdiagramm des Prüfprozesses einer Tabelle bzw. einer Struktur | 26 |
| Darstellung 15: Benutzerschnittstelle mit Protokollausgabe..... | 27 |
| Darstellung 16: Visueller Prototyp der finalen Benutzerschnittstelle | 28 |
| Darstellung 17: Vervollständigte Benutzerschnittstelle | 29 |

1 Einleitung

Zur besseren Lesbarkeit wird in der vorliegenden Arbeit auf die gleichzeitige Verwendung männlicher und weiblicher Sprachformen verzichtet. Es wird das generische Maskulinum verwendet, das stellvertretend für beide Geschlechter steht.

Das erste Kapitel dient einer Einführung in die Thematik dieser Abschlussarbeit. Hierfür werden die Motivation und die Zielsetzung dargestellt. Anschließend wird ein Überblick über den Aufbau gegeben.

1.1 Motivation

Typischerweise wird bei einer Neuentwicklung von SAP-Software auf bereits vorhandenen Softwaremodulen aufgebaut. Um eine Neuentwicklung in einem Kundensystem durchführen zu können, muss ein Dienstleister wie die Dematic GmbH eigenentwickelte Softwaremodule, auf denen die Neuentwicklung basiert, vorab in das Kundensystem exportieren. Zur Erstellung und Verwaltung von derartigen Exporten nutzt der Auftraggeber dieser Arbeit, die Dematic GmbH, eine eigenentwickelte Lösung – das Dematic Project And Transport Management System.

Diese Lösung gibt dem Projektmanagement u. a. die Möglichkeit, den zu exportierenden Projektumfang in Form von Paketen festzulegen. Hierfür werden diese einem SAP-Softwaretransportauftrag zugeordnet. Der Einsatz dieser Lösung bietet – im Gegensatz zum im SAP eingebauten Transportverwaltungssystem – u. a. den Vorteil der automatisierten Dokumentationserstellung, die es ermöglicht, nachvollziehen zu können, welche Pakete bei einem Kunden zum Einsatz gekommen sind. Diese Information kann der Projektleitung u. U. bei der Erstellung von Transporten mit ähnlichen Anforderungen hilfreich sein.

Wenn Entwicklungsobjekte, die den Paketen zugeordnet sind, jedoch Objekte referenzieren, deren Pakete nicht Teil der ausgewählten Paketstruktur sind, würden diese dem Transportauftrag nicht zugeordnet und somit nicht transportiert werden. Die fehlenden Objekte fallen zumeist erst nach dem Import auf. Dieser Umstand führt häufig zu Nachtransporten, deren Erstellung mit vermeidbarem Mehraufwand verbunden ist.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, ein Programm zu entwickeln, welches solche Referenzen auf Objekttypen ermittelt, welche bei der Dematic GmbH für einen Großteil aller Nachtransporte verantwortlich sind. Die ermittelten Objekte sollen in einem Protokoll ausgegeben werden.

Hierfür wird der Transportauftrag durchsucht. Ein Algorithmus prüft daraufhin, ob die zugeordneten Entwicklungsobjekte Objekte referenzieren, die eventuell nicht Bestandteil der von der Projektleitung für das Projekt als notwendig betrachteten Paketstruktur sind. Das Projektmanagement soll vor einem Transport auf derartige Referenzen hingewiesen werden, indem die referenzierten Objekte angezeigt werden.

1.3 Struktur und Aufbau

Im folgenden Kapitel werden die theoretischen Grundlagen behandelt. Es werden Begriffe erläutert, die für das weitere Verständnis dieser Ausarbeitung von Bedeutung sind.

Die Kapitel 3 und 4 umfassen die Anwendungsplanung und -konzeption bzw. die Umsetzung der erarbeiteten Konzepte.

Es folgt eine Zusammenfassung des Planungs- und Entwicklungsprozesses; abschließend werden die erreichten Ergebnisse präsentiert und ein Ausblick über mögliche zukünftige Erweiterungen der entwickelten Anwendung wird gegeben.

2 Theoretische Grundlagen

Das nachfolgende Kapitel dient der Vermittlung der für das Verständnis dieser Arbeit notwendigen theoretischen Grundlagen. Zu diesem Zweck werden die verwendeten Begrifflichkeiten definiert und/oder erläutert.

2.1 Projekt

Ein Projekt ist eine geplante oder bereits begonnene Unternehmung [Du21]. Bei der Dematic GmbH besteht ein Projekt zumeist aus einer Hard- und Softwareentwicklungsaufgabe. Im fortwährenden Verlauf dieser Arbeit bezieht sich dieser Begriff auf die Softwareentwicklung.

2.2 ABAP

Das zu implementierende Programm wird ausschließlich mit der von SAP entwickelten proprietären Programmiersprache ABAP entwickelt. Da diese einige spezielle Merkmale aufweist, die in anderen Programmiersprachen üblicherweise nicht anzutreffen sind, folgt eine Vorstellung der wesentlichsten, für diese Arbeit relevanten Komponenten von ABAP.

2.2.1 ABAP-Dictionary

Das ABAP-Dictionary dient der persistenten Ablage von Typdefinitionen, die in allen Entwicklungsobjekten sichtbar und verwendbar sind [SA21g].

2.2.2 Eingebaute elementare Datentypen

Eingebaute elementare Datentypen sind solche, die in den Sprachumfang von ABAP eingebaut und nicht aus anderen Typen zusammengesetzt sind [SA21l]. Die in dieser Arbeit verwendete Begrifflichkeit *Elementartyp* bezieht sich auf eingebaute elementare Datentypen.

2.2.3 Datenelemente

Mit einem Datenelement des ABAP-Dictionary wird ein elementarer Datentyp oder ein Referenztyp definiert. Zudem wird damit neben den technischen Typeigenschaften auch die semantische Bedeutung eines Objekts beschrieben, das mit Bezug auf das Datenelement definiert wird [SA21j]. Die technischen Eigenschaften eines Datenelements können mittels einer Domäne oder direkt beim Datenelement definiert werden.

2.2.4 Domänen

Domänen sind Entwicklungsobjekte, mit denen technische und semantische Eigenschaften von elementaren Datentypen festgelegt werden. Datenelemente können mit Bezug auf eine Domäne angelegt werden und übernehmen die dort definierten Eigenschaften [SA21k].

2.2.5 Strukturen

Strukturen sind Datenobjekte, die sich aus Komponenten beliebiger, im Speicher hintereinander abgelegter Datentypen (z. B. aus Datenelementen) zusammensetzen [SA21t].

2.2.6 Interne Tabellen

Interne Tabellen dienen der Abspeicherung variabler Datenmengen einer festen Struktur im Arbeitsspeicher von ABAP. Eine interne Tabelle ist ein Datenobjekt, das beliebig viele Zeilen eines beliebigen Datentyps enthalten kann, deren Anzahl nicht statisch festgelegt ist [SA21o].

2.2.7 Tabellentypen

Der Tabellentyp des ABAP-Dictionary dient der Definition des Datentyps einer internen Tabelle in ABAP. Mit dem Tabellentyp werden der Zeilentyp, die Tabellenart und die Tabellenschlüssel einer internen Tabelle festgelegt. Ein programminternes Objekt, das durch Bezug auf einen Tabellentyp deklariert wird, ist eine interne Tabelle dieses Typs [SA21u].

2.2.8 Datenbanktabellen

Datenbanktabellen dienen der persistenten Ablage von Daten. Die Felder einer solchen Tabelle werden durch eine Struktur beschrieben. Für den Zugriff auf die Daten einer Datenbanktabelle stellt ABAP Open-SQL-Anweisungen¹ zur Verfügung.

Die für diese Arbeit relevanten Datenbanktabellen werden nachfolgend vorgestellt:

| Tabellenname | Erklärung |
|--------------|---|
| TADIR | enthält u. a. Objektnamen, Objekttypen und Paketzugehörigkeiten aller Entwicklungsobjekte [SA21f] |

¹ Open-SQL-Anweisungen bilden den DML-Anteil von SQL in ABAP ab, der von allen Datenbanksystemen unterstützt wird [SA21p].

| | |
|-------|---|
| E070 | enthält die Header ² aller Transportaufträge bzw. aller Transportaufgaben [SA21c] |
| E070V | enthält u. a. den Namen des Inhabers und die Kurzbeschreibung von Transportaufträgen [SA21d] |
| E071 | enthält die Entwicklungsobjekte, die einem Transportauftrag bzw. einer Transportaufgabe zugeordnet sind [SA21e] |
| DD03L | enthält die Felder einer Datenbanktabelle bzw. einer Struktur [SA21a] |
| DD40L | enthält die im ABAP-Dictionary definierten Tabellentypen [SA21b] |

Darstellung 1: Erklärung der wesentlichsten, für diese Ausarbeitung relevanten Datenbanktabellen

2.2.9 Dynamische Programme

Dynamische Programme (Dynpros) dienen der Interaktion zwischen dem Nutzer und dem SAP-System. Sie bestehen aus einer grafischen Benutzeroberfläche und einer Ablauflogik.

Custom Controls sind rechteckige Bereiche auf dem Bildschirm eines Dynpros, in die sogenannte Controls eingebettet werden. Bei Controls handelt es sich um Softwarekomponenten des Präsentationsservers, die es ermöglichen, spezifische Tätigkeiten, z. B. das Editieren eines Texts auf dem Präsentationsserver auszuführen [SA21h].

2.2.10 Application Log

Das Application Log ist ein Tool zum Sammeln von Meldungen, Ausnahmen und Fehlern. Diese Informationen werden in einem Protokoll zusammengefasst und dargestellt [SA21r]. In dieser Arbeit wird die Begrifflichkeit *Protokoll* für Application Log verwendet.

2.2.11 Pakete

Pakete kapseln Entwicklungsobjekte in abgeschlossenen Einheiten [SA21q].

² Bei einem Header handelt es sich um eine allgemeine Sammlung von Eigenschaften für einen Transportauftrag bzw. eine Transportaufgabe.

2.2.12 Transportaufträge und Transportaufgaben

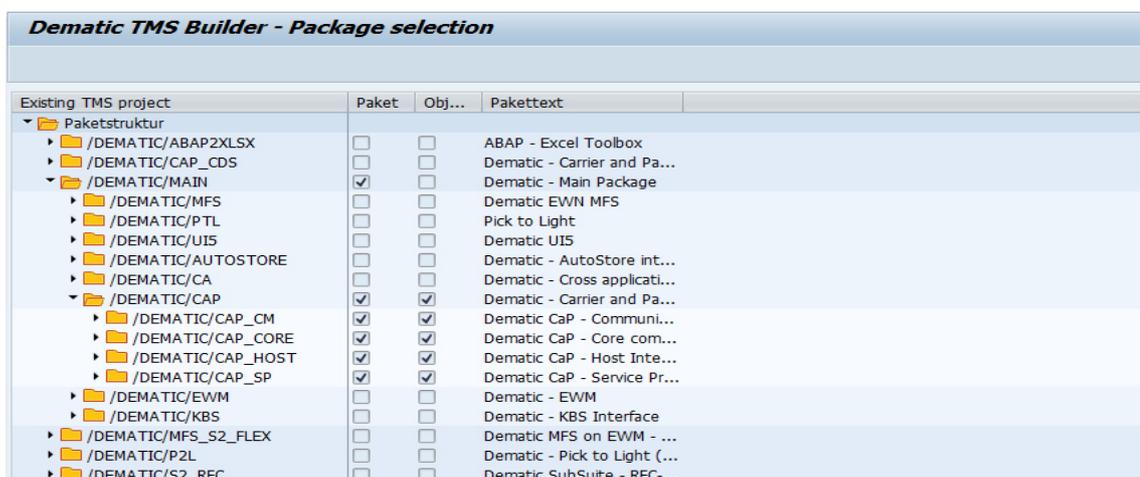
In Transportaufträgen werden Neuentwicklungen und Änderungen an Entwicklungsobjekten zusammengefasst. Derartige Aufträge dienen dazu, diese Zusammenfassung nach Freigabe des Auftrags von einem Quellsystem in ein Zielsystem zu exportieren. Ein Entwicklungsobjekt muss dabei immer einem Transportauftrag zugeordnet sein. Bei einer Neuanlage oder einer Änderung des Entwicklungsobjekts muss dieses einem bestehenden oder neu anzulegenden Auftrag hinzugefügt werden [Eb21]. Hierfür wird eine Transportaufgabe angelegt, die einem SAP-Benutzer zugeordnet ist. Diese Transportaufgabe ist einem Transportauftrag untergeordnet und enthält alle neu angelegten oder geänderten Entwicklungsobjekte eines SAP-Entwicklers.

Zu beachten ist, dass Entwicklungsobjekte Transportaufträgen auch direkt zugeordnet werden können. In diesem Fall ist das entsprechende Objekt keiner Transportaufgabe zugehörig.

2.3 Dematic Project And Transport Management System

Mittels des Dematic Project And Transport Management Systems (Dematic TMS) können Transportaufträge systemübergreifend (z. B. von einem Dematic-SAP-System in ein SAP-Kundensystem) übertragen werden. Das Dematic TMS stellt dem Projektmanagement hierbei verschiedene Werkzeuge zur Projekterstellung und -verwaltung zur Verfügung.

Eines dieser Werkzeuge ist der Dematic Transport Builder (Dematic TB). Damit kann die Projektleitung den an den Kunden auszuliefernden Projektumfang in Form von Paketen festlegen. Hierfür werden den Dematic-Projekten zunächst leere Transportaufträge zugeordnet. Diesen Transportaufträgen werden Pakete aus dem `/DEMATIC/`-Namensraum zugewiesen (vgl. Darstellung 2), die als Grundlage für die spätere Neuentwicklung im Kundensystem dienen sollen [A118].



| Existing TMS project | Paket | Obj... | Pakettext |
|------------------------|-------------------------------------|-------------------------------------|-------------------------------|
| ▼ Paketstruktur | | | |
| ▶ /DEMATIC/ABAP2XLSX | <input type="checkbox"/> | <input type="checkbox"/> | ABAP - Excel Toolbox |
| ▶ /DEMATIC/CAP_CDS | <input type="checkbox"/> | <input type="checkbox"/> | Dematic - Carrier and Pa... |
| ▼ /DEMATIC/MAIN | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Dematic - Main Package |
| ▶ /DEMATIC/MFS | <input type="checkbox"/> | <input type="checkbox"/> | Dematic EWN MFS |
| ▶ /DEMATIC/PTL | <input type="checkbox"/> | <input type="checkbox"/> | Pick to Light |
| ▶ /DEMATIC/UI5 | <input type="checkbox"/> | <input type="checkbox"/> | Dematic UI5 |
| ▶ /DEMATIC/AUTOSTORE | <input type="checkbox"/> | <input type="checkbox"/> | Dematic - AutoStore int... |
| ▶ /DEMATIC/CA | <input type="checkbox"/> | <input type="checkbox"/> | Dematic - Cross applicati... |
| ▼ /DEMATIC/CAP | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Dematic - Carrier and Pa... |
| ▶ /DEMATIC/CAP_CM | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Dematic CaP - Communi... |
| ▶ /DEMATIC/CAP_CORE | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Dematic CaP - Core com... |
| ▶ /DEMATIC/CAP_HOST | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Dematic CaP - Host Inte... |
| ▶ /DEMATIC/CAP_SP | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Dematic CaP - Service Pr... |
| ▶ /DEMATIC/EWM | <input type="checkbox"/> | <input type="checkbox"/> | Dematic - EWM |
| ▶ /DEMATIC/KBS | <input type="checkbox"/> | <input type="checkbox"/> | Dematic - KBS Interface |
| ▶ /DEMATIC/MFS_S2_FLEX | <input type="checkbox"/> | <input type="checkbox"/> | Dematic MFS on EWM - ... |
| ▶ /DEMATIC/P2L | <input type="checkbox"/> | <input type="checkbox"/> | Dematic - Pick to Light (...) |
| ▶ /DEMATIC/S2_RFC | <input type="checkbox"/> | <input type="checkbox"/> | Dematic SubSuite - RFC... |

Darstellung 2: Paketauswahl unter Zuhilfenahme des Dematic TB (eigene Darstellung)

2.4 Paketprüfung

Der im weiteren Verlauf dieser Arbeit verwendete Begriff *Paketprüfung* umfasst folgenden Prozess: das Abgleichen des Pakets eines Objekts mit Paketen, die im Dematic TB ausgewählt wurden. Die Prüfung hat zum Ziel, Objekte zu identifizieren, deren Pakete nicht Teil der für den Export ausgewählten Paketstruktur sind.

3 Anwendungsplanung

Thema dieses Kapitels ist die Anwendungsplanung. Hierfür werden zunächst die Anforderungen an das zu entwickelnde Programm ermittelt. Darauf aufbauend wird ein Vorgehensmodell ausgewählt, um anschließend Konzepte für die Implementierung der Anwendung zu entwerfen.

3.1 Anforderungsanalyse

In einer Befragung der Auftraggeber durch den Verfasser dieser Arbeit konnten folgende Anforderungen ermittelt werden:

- Funktionale Anforderungen:
 - Die Benutzeroberfläche der Anwendung soll in englischer Sprache verfügbar sein.
 - Ein Dynpro soll die Eingabe des zu prüfenden Transportauftrags ermöglichen.
 - Das Dynpro soll Informationen über das dem Transportauftrag zugehörige Projekt darstellen.
 - Der eingegebene Transportauftrag soll auf Datenelemente, Strukturen, Datenbanktabellen und interne Tabellen hin untersucht werden, die Objekte referenzieren, deren Pakete nicht Teil der im Dematic TB ausgewählten Paketstruktur sind.
 - Das Dynpro soll die referenzierten Objekte anzeigen.
 - Der Prüfvorgang soll nachvollziehbar in einem Protokoll festgeschrieben werden.
- Nichtfunktionale Anforderungen:
 - Der Algorithmus soll eine hinnehmbare Geschwindigkeit aufweisen.
 - Das Programm soll um die Prüfung weiterer Objekttypen erweiterbar sein.

Der Verfasser dieser Arbeit hat festgestellt, dass neben den zu untersuchenden Objekten auch andere ABAP-Objekte wie Komponenten von Klassen Objekte referenzieren können, deren Pakete nicht im Dematic TB ausgewählt wurden. In einem Gespräch mit dem Auftraggeber wurde in Erfahrung gebracht, dass solche Objekte bei der Dematic GmbH lediglich in seltenen Ausnahmefällen zu Nachtransporten führen. In dieser Arbeit liegt der Fokus deshalb auf der Paketprüfung der in der Anforderungsanalyse ermittelten Objekttypen.

3.2 Bestimmung des geeignetsten prototypischen Vorgehensmodells

Nachfolgend wird auf Basis der ermittelten Anforderungen und des durch die Hochschule vorgegebenen Abgabetermins, dem 04.03.2022, ein geeignetes prototypisches Vorgehensmodell zur Softwareentwicklung bestimmt. Der Autor dieser Arbeit hat sich für ein prototypisches Vorgehensmodell entschieden, da ein solches dafür geeignet ist, den beiden häufig anzutreffenden Gründen für das Scheitern von IT-Projekten vorzubeugen. Diese sind das Nichterfüllen der Benutzeranforderungen und das Feststellen von technischen Schwierigkeiten in späten Entwicklungsphasen [Wi11]. Zum einen können die erhobenen Anforderungen durch den Verfasser in Absprache mit dem Auftraggeber anhand der Prototypen laufend präzisiert und verifiziert werden. Zum anderen kann der Autor durch die schnellen ersten Ergebnisse, die durch das Anwenden eines prototypischen Vorgehensmodells zu erwarten sind, bereits in frühen Entwicklungsphasen etwaige Probleme in den dem Prototyp zugrundeliegenden Konzepten erkennen.

3.2.1 Vorstellung der gängigsten prototypischen Vorgehensmodelle

Im Folgenden werden die drei gängigsten prototypischen Ansätze [Wi11] vorgestellt. In einer anschließenden Analyse wird daraufhin der optimale Entwicklungsansatz bestimmt.

3.2.1.1 Exploratives Prototyping

Das explorative Prototyping dient der Feststellung der Anforderungen der Anwender. Demonstrationsprototypen³ fungieren hierbei als Grundlage für eine Diskussion zwischen Auftraggebern, Anwendern und Entwicklern bezüglich der Anforderungen an das zu entwickelnde System [Wi11]. Essenziell sind die gemeinsame Diskussion und die Beurteilung verschiedener Lösungsansätze. Diese Art des Prototyping wird bei unklaren Problemstellungen angewandt.

3.2.1.2 Experimentelles Prototyping

Experimentelles Prototyping dient der Beantwortung von Fragen zu Konstruktionsalternativen. Insbesondere wird geprüft, ob die ermittelten Anforderungen technisch umsetzbar sind. Hierfür werden Labormuster eingesetzt, die z. B. verschiedene Architekturalternativen abbilden und dem Entwicklerteam die

³ Ein Demonstrationsprototyp wird für die Auftragsakquise genutzt, diese Art von Prototyp wird nach der Akquisephase verworfen.

grundsätzliche Umsetzbarkeit einer solchen Alternative demonstrieren. Diese Art von Prototyp wird im Allgemeinen nicht zu einem lauffähigen System weiterentwickelt.

3.2.1.3 Evolutionäres Prototyping

Durch diese Art von Prototyping entsteht am Ende ein Pilotsystem, das den Kern des späteren Softwareprodukts darstellt. Hierbei wird der Prototyp inkrementell bis zum endgültigen Produkt weiterentwickelt. Zwischen den Entwicklungszyklen finden ausführliche Tests, Überprüfungen und eine Freigabe durch die Projektbeteiligten statt [Wi11].

3.2.2 Auswahl eines prototypischen Vorgehensmodells

Nachfolgend wird auf Basis der vorgestellten Prototyping-Varianten und der festgestellten Anforderungen der für diese Arbeit geeignetste Entwicklungsansatz ermittelt.

Da die Anforderungen an das zu entwickelnde Softwareprodukt bereits feststehen und die Aufgabenstellung bekannt ist, wird der Ansatz des explorativen Prototyping ausgeschlossen.

Auch das experimentelle Prototyping ist in diesem Kontext ungeeignet, da die zu erstellenden Labormuster Konstruktionsalternativen aufzeigen, diese jedoch nach ihrer Verwendung verworfen werden. Bei dieser Vorgehensweise könnte der von der Hochschule festgelegte Abgabetermin eventuell nicht eingehalten werden.

Das evolutionäre Prototyping hat den Vorteil, dass Zeit und Qualität berücksichtigt werden: Durch die inkrementelle Weiterentwicklung des Softwareprodukts wird auf das zeitintensive Erstellen von Wegwerfprototypen verzichtet. Des Weiteren werden durch fortlaufende Tests und Überprüfungen Fehler schnell und frühzeitig erkannt. Dies beeinflusst nicht nur die Softwarequalität positiv, sondern es hilft auch, eventuelle Wechsel der Lösungsstrategie zügig umsetzen zu können. Zudem kann nach jedem Inkrement eine Überprüfung von diesem durch den Auftraggeber erfolgen. Hierdurch können unerkannte Anforderungen unverzüglich festgestellt und umgesetzt werden.

Aufgrund der erwähnten Vor- und Nachteile der Entwicklungsansätze wird derjenige des evolutionären Prototyping als für diese Arbeit geeignet eingestuft. Dieser führt nicht nur schnell zu ersten überprüfbaren Ergebnissen, deren Qualität durch fortlaufende Modultests sichergestellt wird, sondern ermöglicht auch unter Rücksichtnahme auf eventuelle Verbesserungswünsche der Auftraggeber die inkrementelle Weiterentwicklung des Prototyps zu einem fertigen Softwareprodukt. Der Ansatz des evolutionären Prototyping wird für den gesamten Entwicklungsprozess angewandt.

3.3 Anwendungskonzeption

Auf Grundlage der ermittelten Anforderungen und des gewählten Vorgehensmodells werden nachfolgend Konzepte erarbeitet, die als Grundlage für das zu entwickelnde Programm dienen.

3.3.1 Konzept der Paketprüfung

Die finale Anwendung soll aus möglichst unabhängigen Modulen aufgebaut werden. Aus Prüfungen einfacher Entwicklungsobjekte, die wenige Paketprüfungen benötigen (z. B. Domänen, vgl. hierzu nachfolgend Fall 1), soll so schrittweise, aufbauend auf den bereits gewonnenen Erkenntnissen, eine Prüfung komplexerer Datenstrukturen (z. B. von Strukturen, vgl. hierzu nachfolgend Fall 3) erfolgen. Ein solcher Aufbau hat zum Vorteil, dass der Suchbereich für Fehler auf einzelne Module eingeschränkt und das Programm durch weitere Module ohne Komplikationen erweitert werden kann (vgl. Abschnitt 3.1). Ein Modul soll einen bzw. mehrere ähnliche Prüffälle repräsentieren.

Insgesamt lassen sich fünf Prüffälle unterscheiden, jeder – mit Ausnahme von Fall 3 und 4 – stellt ein eigenes Modul dar:

Fall 1 – Prüfung von Domänen

Da Datenelemente mit Bezug auf Domänen definiert werden können, müssen auch Domänen, zusätzlich zu den in Abschnitt 3.1 genannten zu prüfenden Objekten, einer Paketprüfung unterzogen werden.

In Domänen werden die technischen und semantischen Eigenschaften von elementaren Typen definiert. Deshalb muss lediglich die Domäne einer Paketprüfung unterzogen werden.

Fall 2 – Prüfung von Datenelementen

Mithilfe eines Datenelements wird ein elementarer Datentyp oder ein Referenztyp definiert. Die Typeigenschaften eines elementaren Typs können entweder direkt beim Datenelement angegeben oder aus einer Domäne übernommen werden. Werden diese Eigenschaften direkt angegeben, muss die Paketprüfung ausschließlich auf das Datenelement angewandt werden. Werden die Typeigenschaften jedoch aus einer Domäne übernommen, müssen die Domäne und das Datenelement einer Paketprüfung unterzogen werden.

Mit einem Datenelement kann zudem ein Referenztyp beschrieben werden. Aufgrund der Seltenheit dieses Falles wurde in Absprache mit dem Auftraggeber entschieden, kein Prüfprozedere für einen solchen Fall zu implementieren.

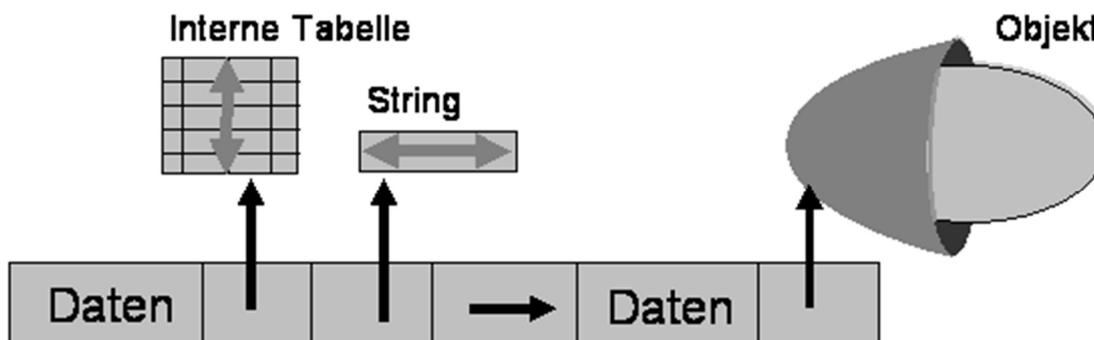
Fall 3 – Prüfung von Strukturen

Eine Struktur kann aus Komponenten beliebiger anderer Datentypen bestehen. Hierbei lassen sich vier Arten von Strukturen unterscheiden:

- geschachtelte und nicht geschachtelte Strukturen sowie
- flache und tiefe Strukturen.

Nicht geschachtelte Strukturen enthalten keine weiteren Unterstrukturen, während geschachtelte Strukturen mindestens eine Unterstruktur umfassen.

Flache Strukturen beinhalten ausschließlich Datentypen fester Länge (einschließlich Unterstrukturen mit Datentypen fester Länge), wohingegen tiefe Strukturen Komponenten wie Referenztypen, Strings oder interne Tabellen enthalten (vgl. Darstellung 3). Diese Komponenten können – mit Ausnahme von Strings – wiederum weitere Verschachtelungsebenen aufweisen [Ja09].



Darstellung 3: Beispielhafter Aufbau einer tiefen Struktur [SA21t]

Von den genannten Strukturen stellen nicht geschachtelte flache Strukturen und nicht geschachtelte tiefe Strukturen den einfachsten Prüffall dar. Die Paketprüfung muss bei einer nicht geschachtelten flachen Struktur auf die Struktur und auf die einzelnen Komponenten dieser Struktur angewandt werden. Im Falle einer nicht geschachtelten tiefen Struktur müssen zudem eventuell vorhandene interne Tabellen und die referenzierten Objekte eines bzw. mehrerer Referenztypen kontrolliert werden, um eine vollständige Paketprüfung aller Komponenten zu gewährleisten.

Geschachtelte Strukturen erfordern neben der Paketprüfung der Struktur und der einzelnen Strukturkomponenten zudem eine Prüfung aller Komponenten der Unterstrukturen.

Fall 4 – Prüfung von Datenbanktabellen

Die Felder einer Datenbanktabelle werden durch eine Struktur beschrieben. Diese muss nicht geschachtelt sein [SA21i]. Die Paketprüfung wird daher auf die Datenbanktabelle und – wie in Fall 3 bereits erwähnt – auf die Struktur sowie die einzelnen Komponenten dieser angewandt. Da die Paketprüfung von Datenbanktabellen maßgeblich von der Prüfung dieser Struktur abhängt, werden Fall 4 und Fall 3 in einem Modul zusammengefasst.

Fall 5 – Prüfung von Tabellentypen

Da mit dem gewählten Tabellentyp der Zeilentyp interner Tabellen festgelegt wird und interne Tabellen wiederum Komponenten von Strukturen darstellen können, müssen auch Tabellentypen einer Paketprüfung unterzogen werden. Dies erfolgt zusätzlich zu den in Abschnitt 3.1 genannten Objekten.

Für diese Arbeit ist der durch den Tabellentyp festgelegte Zeilentyp von besonderer Bedeutung. Dieser kann durch einen elementaren Datentyp, durch die direkte Angabe der technischen Eigenschaften eines eingebauten Typs, durch einen Referenztyp, eine Domäne, eine Struktur oder durch einen weiteren Tabellentyp definiert werden. Die Paketprüfung muss immer auf den Tabellentyp und – je nach Zeilentyp – auch auf den Zeilentyp angewandt werden. Nachfolgend werden die unterschiedlichen Prüffälle zusammen mit den für diese Fälle vorgesehenen Prüfkonzepten vorgestellt:

Fall 5a:

Der Zeilentyp wird durch einen elementaren Typ definiert oder durch direkte Eingabe eines eingebauten Typs festgelegt. Eine Prüfung des Zeilentyps ist nicht erforderlich, da die festlegenden Komponenten in jedem anderen SAP-System zur Verfügung stehen und somit nicht transportiert werden müssen.

Fall 5b:

Der Zeilentyp wird durch eine Domäne oder eine Struktur bestimmt. Um die Paketzugehörigkeit des festlegenden Objekts zu prüfen, wird die unter Fall 1 bzw. Fall 3 beschriebene Prüflogik angewandt.

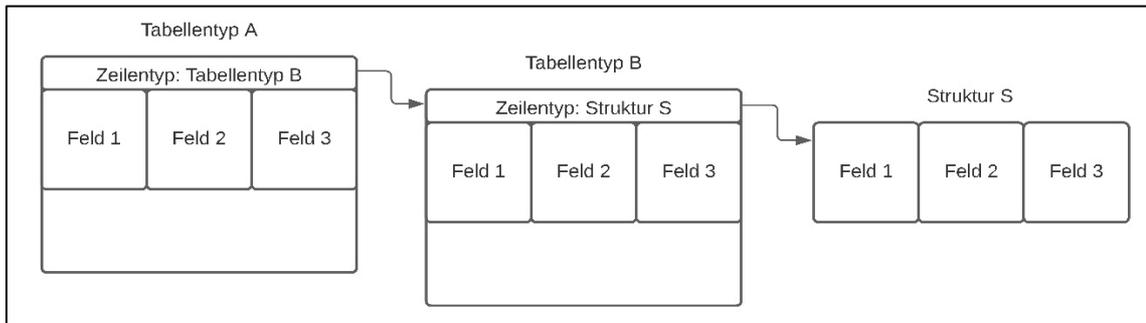
Fall 5c:

Der Zeilentyp wird durch eine Referenz festgelegt. Auf Basis des referenzierten Typs muss entschieden werden, ob und wie eine Paketprüfung zu erfolgen hat.

Referenzierte Klassen und Schnittstellen werden auf Wunsch des Auftraggebers auf oberster Ebene hinsichtlich ihrer Paketzugehörigkeit geprüft. Eine weitere Prüfung, wie sie z. B. auf Komponenten von Klassen angewandt werden könnte, ist vom Auftraggeber nicht gewünscht.

Fall 5d:

Der Zeilentyp wird durch einen weiteren Tabellentyp beschrieben. In diesem seltenen Fall muss die Paketprüfung auf ebenjenen angewandt werden. Wird der Zeilentyp des geprüften Tabellentyps erneut durch einen Tabellentyp festgelegt, muss auch dieser einer Paketprüfung unterzogen werden – dieser Prozess muss solange wiederholt werden, bis der Zeilentyp durch eine andere Datenstruktur (z. B. durch eine Struktur) beschrieben wird. Anschließend erfolgt die Prüfung der Paketzugehörigkeit der Datenstruktur, mit der die Zeilen des letzten ermittelten Tabellentyps beschrieben werden. Dieser Fall wird nachfolgend anhand der Darstellung 4 erläutert.

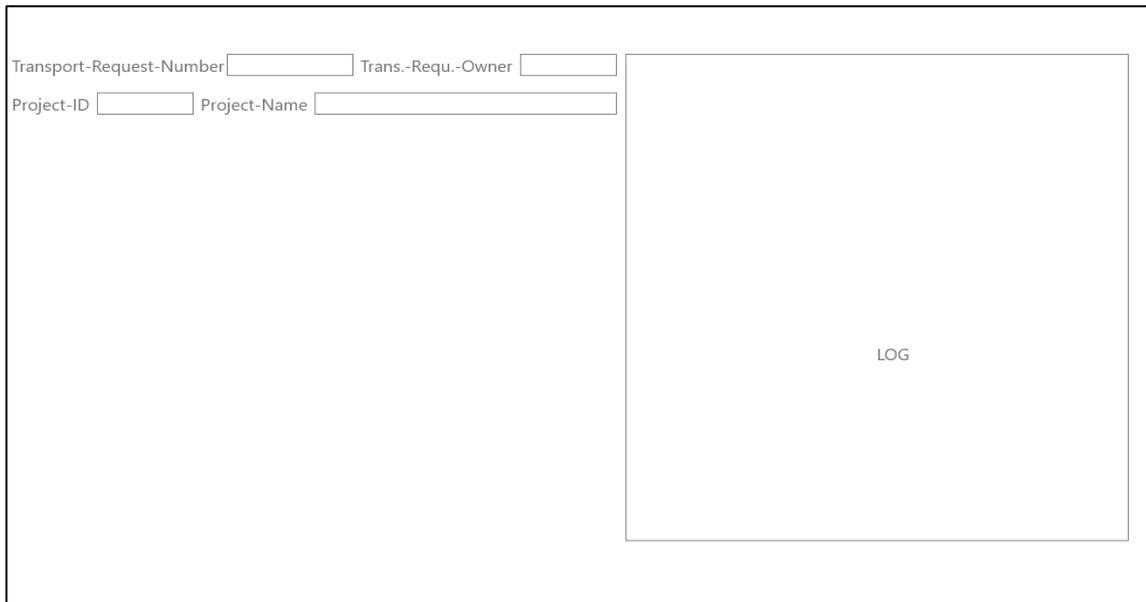


Darstellung 4: Beispielhafter Aufbau eines Tabellentyps (eigene Darstellung)

Wie der obigen Darstellung zu entnehmen ist, wird der Zeilentyp des Tabellentyps A durch den Tabellentyp B festgelegt, dessen Zeilentyp wiederum durch die Struktur S definiert wird. Gemäß des in Fall 5d erläuterten Prüfkonzepts muss somit die Paketzugehörigkeit des Tabellentyps A, des Tabellentyps B und der Struktur S zusammen mit den Komponenten geprüft werden.

3.3.2 Konzept der Benutzerschnittstelle

Die Benutzerschnittstelle ermöglicht dem Anwender die Eingabe des zu prüfenden Transportauftrags. Zugleich sind ihr Informationen bezüglich des dem Transportauftrag zugeordneten Projekts und der ermittelten problematischen Referenzen zu entnehmen. Das in der Darstellung 5 abgebildete Modell stellt einen frühen konzeptionellen Entwurf der Schnittstelle dar.



Darstellung 5: Konzept der Benutzerschnittstelle (eigene Darstellung)

3.3.3 Modultestkonzept

Nach der Fertigstellung eines jeden Anwendungsmoduls werden diverse Modultests durchgeführt, um mögliche Fehler bereits frühzeitig erkennen zu können. Diese Vorgehensweise weist den Vorteil auf, dass der Suchbereich für Fehler klein gehalten werden kann.

4 Umsetzung

Das vierte Kapitel beinhaltet die Implementierung der Anwendung, d. h. die Umsetzung der in Abschnitt 3.3 erarbeiteten Konzepte. Es werden zudem Modultestergebnisse und die Lösung ermittelter Fehler vorgestellt. Außerdem wird nach jedem Modul das Prüfprozedere in Form eines Flussdiagramms abgebildet. Dabei wird jeweils das Diagramm erläutert, um den Prüfprozess verständlicher zu gestalten.

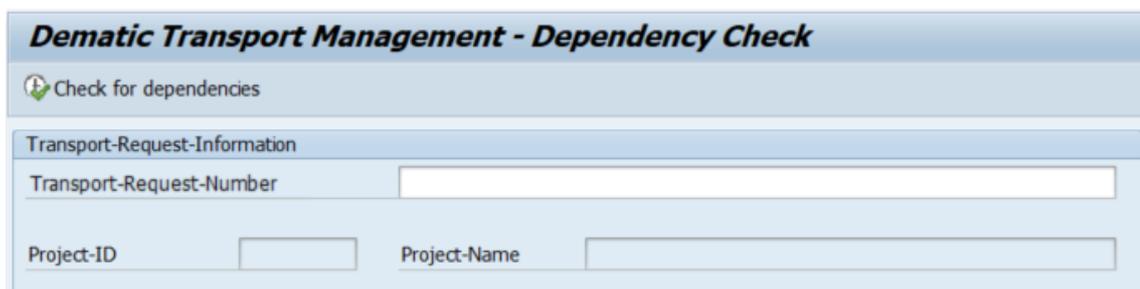
4.1 Prüfung von Domänen

Das erste Modul deckt den einfachsten Fall, die Prüfung der Paketzugehörigkeit von Domänen, ab. Zudem werden die ersten Ein- und Ausgabefelder der Benutzerschnittstelle (vgl. Abschnitt 3.3.2) implementiert. Diese Schnittstelle ermöglicht dem Anwender die Eingabe des zu prüfenden Transportauftrags und enthält Informationen zu dem zum Transportauftrag zugehörigen Projekt (vgl. Abschnitt 2.3). Des Weiteren werden in diesem Modul Basisfunktionalitäten (z. B. das Abfragen des Pakets eines Objekts) implementiert, die in darauffolgenden Modulen benötigt werden.

4.1.1 Die Benutzerschnittstelle – Eingabemaske

Zunächst werden die Eingabemaske des Dynpros und die hierfür benötigten Komponenten implementiert. Die im Konzept vorgesehene Protokollausgabe wird inklusive ihrer Filterfunktionalität in Abschnitt 4.5.1 umgesetzt.

Als Grundlage für die Benutzerschnittstelle dient das vom Verfasser angelegte Programm `/DEMATIC/TMS_CHECK`. Mit diesem kann ausschließlich das Dynpro aufgerufen werden, das der Interaktion mit dem Anwender dient.



Darstellung 6: Eingabemaske des Dynpros zur Abfrage der Nummer des Transportauftrags (eigene Darstellung)

Nach dem Ausführen von `/DEMATIC/TMS_CHECK` wird dem Anwender die Eingabemaske angezeigt, die der Darstellung 6 zu entnehmen ist. Im Folgenden werden die Felder dieser Maske vorgestellt:

Das Eingabefeld *Transport-Request-Number* ermöglicht dem Nutzer, die Nummer des zu prüfenden Transportauftrags zu hinterlegen. Nach der Betätigung der Drucktaste *Check for dependencies* werden mittels einer Datenbankabfrage die dem Transportauftrag zugehörige Projekt-ID und der dem Transportauftrag zugehörige

Projektname ermittelt. Diese Angaben werden in den Ausgabefeldern *Project-ID* und *Project-Name* dargestellt, um den Anwender über das Projekt zu informieren, das dem Auftrag zugeordnet ist. Im Anschluss erfolgt der Prüfvorgang. Hierfür wird die im folgenden Abschnitt vorgestellte öffentliche Prüfmethode der Serviceklasse aufgerufen, mit der wiederum die jeweiligen Paketprüfungen durchgeführt werden.

4.1.2 Die Serviceklasse

Die Klasse */DEMATIC/CL_TMS_CHECK_SRV* stellt die benötigten Grundfunktionalitäten für die gesamte Anwendung zur Verfügung. Im Folgenden werden die im ersten Modul implementierten Methoden vorgestellt.

4.1.2.1 GET_TRANSPORT_TASKS_FOR_REQU

Diese Methode dient der Abfrage aller Transportaufgaben eines gegebenen Transportauftrags. Zu diesem Zweck werden die Felder *TRKORR* und *STRKORR* aus der Tabelle *E070* selektiert. Das erstgenannte Feld enthält die Nummer der Transportaufgabe, dem anderen Feld ist die Nummer des zugeordneten Transportauftrags zu entnehmen.

4.1.2.2 GET_TRANSPORT_OBJECTS_FOR_TASK

Indem die *GET_TRANSPORT_OBJECTS_FOR_TASK*-Methode angewandt wird, wird die *E071*-Tabelle selektiert und alle Transportobjekte, die zu den zuvor ermittelten Transportaufgaben gehören, werden bestimmt.

4.1.2.3 GET_TRANSPORT_OBJECTS_FOR_REQU

Aufbauend auf den beiden vorherigen Methoden werden mithilfe von *GET_TRANSPORT_OBJECTS_FOR_REQU* alle Transportobjekte ermittelt, die zu einem Transportauftrag gehören. Diese werden anschließend in dem Klassenattribut *MT_TRANSPORT_OBJECTS* gespeichert. Dadurch ist es möglich, die ermittelten Objekte in den entsprechenden Prüfmethode der Module abzufragen.

4.1.2.4 GET_RELEVANT_OBJECTS

Bei Anwendung dieser Methode werden alle Transportobjekte gelöscht, die für die Paketprüfung nicht infrage kommen (vgl. Abschnitt 3.3.1). Nach dem Löschvorgang bleiben ausschließlich Domänen, Datenelemente, Strukturen, Tabellen und Tabellentypen erhalten.

4.1.2.5 GET_SELECTED_PACKAGES_FOR_PROJ

Mit *GET_SELECTED_PACKAGES_FOR_PROJ* werden mithilfe der Transportauftragsnummer und des dazugehörigen Projekts (vgl. Abschnitt 2.3) alle Pakete ermittelt, die im Dematic TB für den Export in das Kundensystem ausgewählt wurden.

4.1.2.6 GET_PACKAGE_OF_OBJECT

Mit `GET_PACKAGE_OF_OBJECT` wird durch Abfragen der `TADIR`-Tabelle das Paket eines übergebenen Objekts herausgefunden.

4.1.2.7 CHECK_PACKAGES_OF_DOMAINS

Diese Prüfmethode stellt den Kern des ersten Moduls dar. Mit ihr wird geprüft, ob die für die Domänen des `/DEMATIC/`-Namensraums ermittelten Pakete im Dematic TB für den Export in das Zielsystem ausgewählt wurden. Das Ergebnis wird im Protokoll ausgegeben.



| Type | Message | Text |
|------|---|---|
| | ----- | Checking for dependencies in Domains----- |
| ▲ | Package for domain /DEMATIC/AS_BEGINGROUP was not selected for export. Missing Pckg.: /DEMATIC/AS_CA | |
| ● | No package and TADIR entry was found for Object /DEMATIC/AS_BEGIN_END_GROUP. | |
| ▲ | Package for domain /DEMATIC/AS_DEPTH was not selected for export. Missing Pckg.: /DEMATIC/AS_CA | |
| ▲ | Package for domain /DEMATIC/AS_ENDGROUP was not selected for export. Missing Pckg.: /DEMATIC/AS_CA | |
| ▲ | Package for domain /DEMATIC/AS_ESTPICKTIME was not selected for export. Missing Pckg.: /DEMATIC/AS_CA | |
| ▲ | Package for domain /DEMATIC/AS_GRID_POSITION was not selected for export. Missing Pckg.: /DEMATIC/AS_CA | |
| ▲ | Package for domain /DEMATIC/AS_IS_READY was not selected for export. Missing Pckg.: /DEMATIC/AS_CA | |
| ▲ | Package for domain /DEMATIC/AS_SUSPEND_TASKGROUP was not selected for export. Missing Pckg.: /DEMATIC/AS_CA | |
| ▲ | Package for domain /DEMATIC/AS_TARGET_GRID was not selected for export. Missing Pckg.: /DEMATIC/AS_CA | |
| ▲ | Package for domain /DEMATIC/AS_TASKCOMPLETE was not selected for export. Missing Pckg.: /DEMATIC/AS_CA | |
| ■ | Package for domain /DEMATIC/CAP_DACH_BRANCH_NR was selected for export. Package: /DEMATIC/CAP_SP_DACH. | |
| ■ | Package for domain /DEMATIC/CAP_DACH_EDIFACT was selected for export. Package: /DEMATIC/CAP_SP_DACH. | |

Darstellung 7: Beispielhaftes Ergebnis einer Paketprüfung von Domänen (eigene Darstellung)

Der Darstellung 7 ist eine durch die Prüfmethode generierte Protokollausgabe zu entnehmen. Erkennbar sind gelbe Warnungen, die Objekte anzeigen, deren Pakete nicht im Dematic TB ausgewählt wurden. Beispielsweise wurde das Paket `/DEMATIC/AS_CA`, in dem die Domäne `/DEMATIC/AS_BEGINGROUP` gekapselt wurde, nicht für den Export ausgewählt. Die rote Fehlermeldung zeigt an, dass für ein Objekt kein Paket ermittelt werden konnte. Durch diese Meldung soll die Projektleitung darauf hingewiesen werden, dass das entsprechende Objekt manuell geprüft werden muss. Die grünen Informationen signalisieren, dass die Domäne kontrolliert und das Paket dieser Domäne im Dematic TB für den Export ausgewählt wurde.

4.1.2.8 CHECK_ALL_OBJECTS

Als einzige öffentliche Methode der Serviceklasse dient die `CHECK_ALL_OBJECTS`-Methode dazu, die Paketprüfung der zu untersuchenden Objekte vollständig durchzuführen. Dies bedeutet, dass bei ihrer Anwendung alle benötigten Prüfmethoden (vgl. Abschnitt 3.3.1) aufgerufen werden.

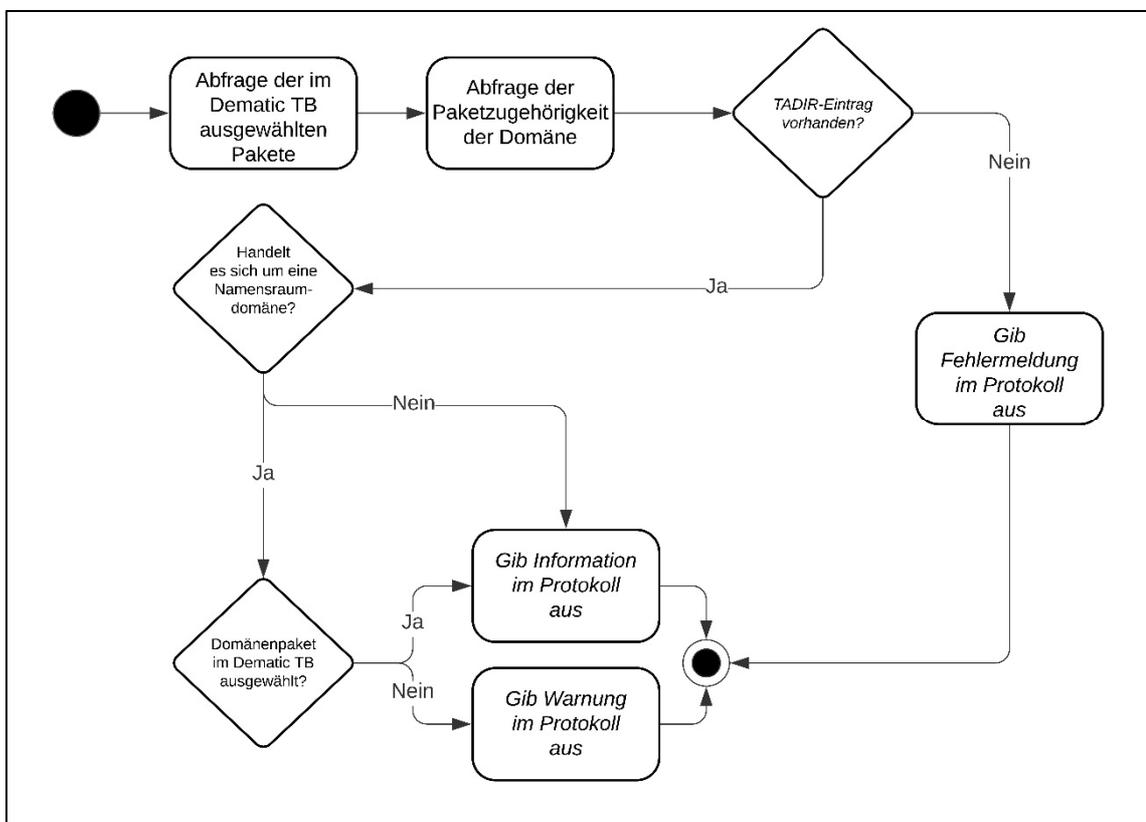
4.1.3 Modultestergebnis

Die durchgeführten Modultests verliefen im Allgemeinen positiv. Jedoch stellte sich heraus, dass von der SAP erstellte Domänen wie `CHAR4` als nicht für den Export vorgesehen erkannt wurden und es für einige aus den Transportaufträgen selektierte

Domänen keine *TADIR*-Einträge gab (z. B. weil die zu untersuchende Domäne gelöscht wurde).

Das erstgenannte Problem wird durch eine Einschränkung des Prüfalgorithmus behoben. Dieser prüft anwendungsweit nun nur noch Objekte, die dem */DEMATIC/*-Namensraum angehören, da von der SAP erstellte Objekte in jedem anderen SAP-System zur Verfügung stehen und somit nicht transportiert werden müssen. Für das zweite Problem wird eine Protokollausgabe implementiert, anhand derer kenntlich wird, dass das entsprechende Objekt keiner Prüfung unterzogen werden konnte. Dies ist darauf zurückzuführen, dass für dieses – mangels *TADIR*-Eintrags (vgl. Abschnitt 4.1.2.6) – kein Paket ermittelt werden konnte.

4.1.4 Übersicht über den Prüfprozess einer Domäne



Darstellung 8: Flussdiagramm des Prüfprozesses einer Domäne (eigene Darstellung)

Im Prüfprozess einer Domäne werden zunächst die im Dematic TB für den Export ausgewählten Pakete selektiert, anschließend das Paket der zu prüfenden Domäne. Kann kein Domänenpaket ermittelt werden (d. h., es ist kein *TADIR*-Eintrag vorhanden), wird der Prüfprozess abgebrochen; dies wird im Protokoll mittels einer roten Fehlermeldung vermerkt. Wurde für die Domäne ein Paket ermittelt, wird daraufhin geprüft, ob es sich bei dieser Domäne um eine Domäne des */DEMATIC/*-Namensraums handelt. Ist dies nicht der Fall, wird der Prüfprozess ebenfalls abgebrochen, da es sich um eine von der SAP erstellte Domäne handelt. Dies wird auch im Protokoll vermerkt.

Handelt es sich um eine von der Dematic erstellte Domäne, folgt die eigentliche Paketprüfung: Wurde das Paket der Domäne im Dematic TB ausgewählt, wird eine Information diesbezüglich im Protokoll ausgegeben. Andernfalls wird eine Warnung vermerkt, mit der die Projektleitung darüber informiert wird, dass das Paket der Domäne nicht für den Export ausgewählt wurde.

Der beschriebene Prozess wird – leicht verändert – auch in späteren Flussdiagrammen verwendet. Um Redundanzen zu vermeiden, wird dieser Prozess in einer neuen Aktivität, der Aktivität *Paketprüfung*, zusammengefasst. Diese beinhaltet alle in der Darstellung 8 enthaltenen Aktivitäten, angepasst an die zu prüfende Datenstruktur. Beispielsweise umfasst die Aktivität *Paketprüfung des Datenelements* in der Darstellung 10 anstelle der Aktivität *Abfrage der Paketzugehörigkeit der Domäne* die Aktivität *Abfrage der Paketzugehörigkeit des Datenelements*.

4.2 Prüfung von Datenelementen

Das zweite Modul beinhaltet die Prüfung der Paketzugehörigkeit von Datenelementen. Um ein solches zu prüfen, werden in der Serviceklasse (vgl. Abschnitt 4.1.2) zwei neue Methoden implementiert. Diese werden im Folgenden präzisiert:

Mithilfe der *GET_DOMAIN_OF_DATA_ELEMENT*-Methode wird auf Basis eines zu übergebenen Datenelements die zugehörige Domäne ermittelt. Hierbei werden zwei Fälle berücksichtigt (vgl. Abschnitt 3.3.1, Fall 2):

Fall 1:

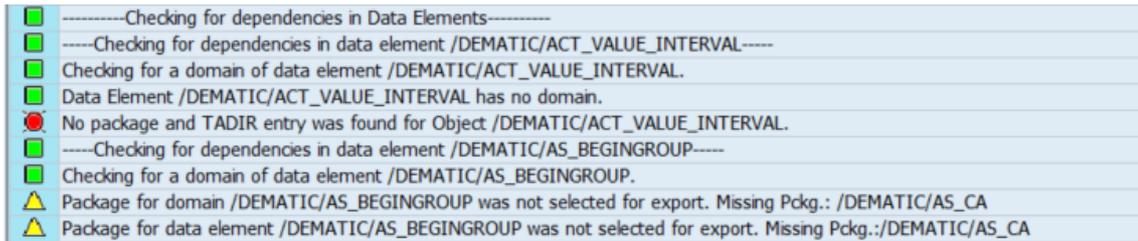
Die technischen Eigenschaften des Datenelements werden durch direkte Eingabe beschrieben, d. h., dem Datenelement ist keine Domäne zugehörig. In diesem Fall wird eine Information im Protokoll ausgegeben, mit der darauf hingewiesen wird, dass keine Domänenprüfung zu erfolgen hat.

Fall 2:

Werden die technischen Eigenschaften des Datenelements durch eine Domäne beschrieben, wird diese hinsichtlich ihrer */DEMATIC/-*Namensraumzugehörigkeit geprüft. Verläuft dieser Test positiv, wird die ermittelte Domäne an den Aufrufer zurückgegeben. Andernfalls wird eine Information im Protokoll ausgegeben, der zu entnehmen ist, dass es sich um eine von der SAP erstellte Domäne handelt.

Die Paketprüfung aller Datenelemente eines Transportauftrags wird mit der *CHECK_PACKAGES_OF_DATA_ELEMENT*-Methode realisiert. Durch diese Methode werden alle Datenelemente schrittweise dahingehend geprüft, ob ihnen eine Domäne zugehörig ist. Die Prüfung nach einer zugehörigen Domäne wird durch einen Aufruf der *GET_DOMAIN_OF_DATA_ELEMENT*-Methode verwirklicht. Wurde eine Domäne ermittelt, wird die in Abschnitt 4.1.2.7 beschriebene Methode zur Überprüfung der Paketzugehörigkeit einer Domäne aufgerufen. Anschließend werden die */DEMATIC/-*

Namensraumprüfung und die Prüfung der Paketzugehörigkeit des Datenelements durchgeführt. Die Ergebnisse werden im Protokoll festgehalten.



Darstellung 9: Beispielhaftes Ergebnis einer Paketprüfung von Datenelementen (eigene Darstellung)

Der Darstellung 9 ist eine durch die `CHECK_PACKAGES_OF_DATA_ELEMENT`-Methode generierte Protokollausgabe zu entnehmen. Diese stellt den Prüfprozess der Datenelemente `/DEMATIC/ACT_VALUE_INTERVAL` und `/DEMATIC/AS_BEGIN_GROUP` dar.

Für das erstgenannte Datenelement konnte keine Domäne ermittelt werden. Zudem konnte mangels `TADIR`-Eintrags kein Paket bestimmt werden. Die Fehlermeldung (rot) weist darauf hin, dass das Datenelement aus diesem Grund keiner Paketprüfung unterzogen werden konnte.

Für das Datenelement `/DEMATIC/AS_BEGINGROUP` wurden hingegen eine Domäne und ein Paket ermittelt. Die gelben Warnungen verdeutlichen, dass weder das Paket der Domäne noch dasjenige des Datenelements im Dematic TB ausgewählt wurden.

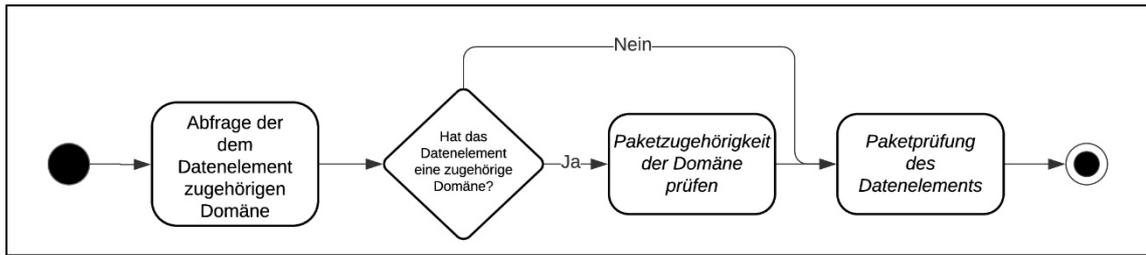
4.2.1 Modultestergebnis

Folgender Fehler trat bei den Modultests auf:

Es stellte sich heraus, dass die Prüfung der Paketzugehörigkeit einer Domäne mehrfach durchgeführt wurde. Daraus resultierte ein Testergebnis von über 70 000 Protokollausgaben anstatt der erwarteten 15 000. Des Weiteren verlängerte sich der Prüfprozess um etwa zwei Minuten.

Der Fehler ist auf ein vergessenes Löschen bereits geprüfter Domänen zurückzuführen: Wird für ein Datenelement eine Domäne ermittelt, so wird, um diese hinsichtlich ihrer Paketzugehörigkeit zu prüfen, die in Abschnitt 4.1.2.7 beschriebene Methode zur Prüfung von Domänen aufgerufen. Zur Übergabe der zu prüfenden Domäne wird diese dem Klassenattribut `MT_TRANSPORT_OBJECTS` (vgl. Abschnitt 4.1.2.3) angehängt, aus dem diese Domäne von der Prüfmethode für den Prüfvorgang ausgelesen wird. Enthält das Attribut weitere Domänen (z. B. aus früheren Prüfvorgängen), so werden diese allesamt einer (wiederholten) Prüfung unterzogen. Um den Fehler zu beheben, werden nun anwendungsweit bereits geprüfte Objekte aus dem Klassenattribut gelöscht.

4.2.2 Übersicht über den Prüfprozess eines Datenelements



Darstellung 10: Flussdiagramm des Prüfprozesses eines Datenelements (eigene Darstellung)

Wie der Darstellung 10 zu entnehmen ist, muss für den Prüfprozess eines Datenelements zunächst untersucht werden, ob dem Datenelement eine Domäne zugehörig ist. Ist dies der Fall, wird diese Domäne zunächst mit der in Abschnitt 4.1.2.7 implementierten Prüfmethode für Domänen hinsichtlich ihrer Paketzugehörigkeit geprüft. Andernfalls, bzw. nach dem Prüfvorgang einer eventuell vorhandenen Domäne, wird die Paketprüfung des Datenelements durchgeführt. Hierfür wird die in Abschnitt 4.1.4 dargestellte Prüflogik mit den dort erläuterten Anpassungen verwendet.

4.3 Prüfung von Tabellentypen

Diesem Modul ist der unter Abschnitt 3.3.1 aufgeführte Fall 5 – die Prüfung von Tabellentypen – implementiert. Zur Prüfung eines solchen Falles wird eine neue Methode angelegt: Mittels der *CHECK_PACKAGES_OF_TABLE_TYPE*-Methode wird die Paketzugehörigkeit von Tabellentypen geprüft.

Hierfür werden zunächst die Eigenschaften des zu untersuchenden Tabellentyps aus der *DD40L*-Tabelle selektiert. Von Bedeutung sind die selektierten Tabellenfelder *ROWKIND* und *TTYPKIND*. Im erstgenannten Feld wird der verwendete Zeilentyp angegeben. Es kann sich z. B. um einen elementaren Typ, einen Referenztyp oder um eine Domäne handeln. Im zweiten Feld ist der Typ des Tabellentyps definiert. Dabei kann es sich entweder um einen allgemeinen Tabellentyp oder um einen Ranges-Tabellentyp⁴ handeln. Um die Komponenten des Zeilentyps eines Ranges-Tabellentyps hinsichtlich ihrer Paketzugehörigkeit zu prüfen, wird die in Abschnitt 4.4 implementierte Prüfmethode (*CHECK_PACKAGES_OF_STRUC_TABL*) aufgerufen.

Das Prüfprozedere für allgemeine Tabellentypen ist aufgrund der Vielzahl der möglichen Zeilentypen (vgl. Abschnitt 3.3.1, Fall 5) umfangreicher und wird nachfolgend erläutert. Der Zeilentyp eines allgemeinen Tabellentyps kann durch einen anderen Tabellentyp beschrieben werden (vgl. hierzu auch Darstellung 4). Ist dies der Fall, wird nach der Prüfung der Paket- und der Namensraumzugehörigkeit des zu untersuchenden Tabellentyps die Paketzugehörigkeit des Tabellentyps geprüft, der den Zeilentyp des zu

⁴ Bei einem Ranges-Tabellentyp handelt es sich um eine interne globale Tabelle. Sie besteht aus vier Feldern und dient der Ablage einer Selektionsbedingung [SA21s].

untersuchenden Tabellentyps beschreibt. Dieser Prozess muss solange wiederholt werden, bis die Zeilen eines solchen durch eine andere Datenstruktur (z. B. durch eine Struktur) beschrieben werden. Nach der Iteration erfolgt die Prüfung der Paketzugehörigkeit der Datenstruktur, mit der die Zeilen des letzten ermittelten Tabellentyps beschrieben werden (vgl. hierzu die nachfolgenden Fälle 1 bis 3). Die ermittelten Prüfergebnisse werden im Protokoll festgeschrieben.

Insgesamt lassen sich fünf weitere Prüffälle unterscheiden. Da sich einige in ihrer Behandlung ähneln, werden diese zu drei Fällen zusammengefasst:

Fall 1:

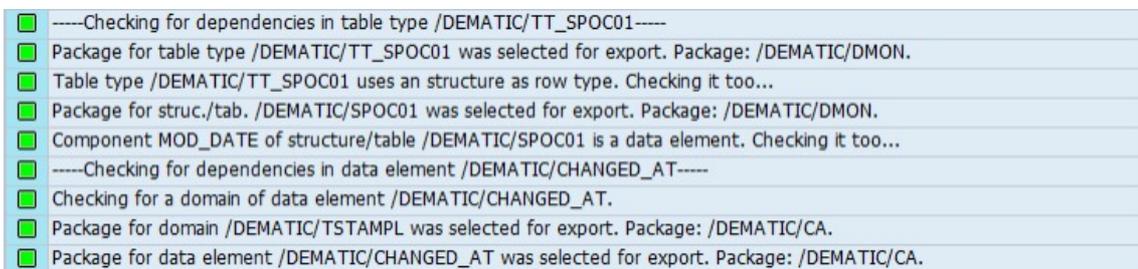
Der Zeilentyp wird mittels eines eingebauten elementaren Typs oder durch direkte Eingabe festgelegt: Es werden Informationen im Protokoll ausgegeben, denen zu entnehmen ist, dass die festlegenden Komponenten in jedem anderen SAP-System zur Verfügung stehen und somit nicht transportiert werden müssen.

Fall 2:

Der Zeilentyp wird durch eine Struktur oder eine Domäne beschrieben. Um diese zu prüfen, wird die Prüfmethode *CHECK_PACKAGES_OF_STRUC_TABL* (vgl. Abschnitt 4.4) bzw. *CHECK_PACKAGES_OF_DOMAINS* (vgl. Abschnitt 4.1.2.7) aufgerufen.

Fall 3:

Der Typ, durch den die Zeile festgelegt wird, ist ein Referenztyp. Zur Prüfung eines solchen Typs wird die unter Abschnitt 4.4, Fall 3 beschriebene Prüfmethode eingesetzt.



```

[ ] ----Checking for dependencies in table type /DEMATI/TT_SPOC01----
[ ] Package for table type /DEMATI/TT_SPOC01 was selected for export. Package: /DEMATI/DMON.
[ ] Table type /DEMATI/TT_SPOC01 uses an structure as row type. Checking it too...
[ ] Package for struc./tab. /DEMATI/SPOC01 was selected for export. Package: /DEMATI/DMON.
[ ] Component MOD_DATE of structure/table /DEMATI/SPOC01 is a data element. Checking it too...
[ ] ----Checking for dependencies in data element /DEMATI/CHANGED_AT----
[ ] Checking for a domain of data element /DEMATI/CHANGED_AT.
[ ] Package for domain /DEMATI/TSTAMPL was selected for export. Package: /DEMATI/CA.
[ ] Package for data element /DEMATI/CHANGED_AT was selected for export. Package: /DEMATI/CA.

```

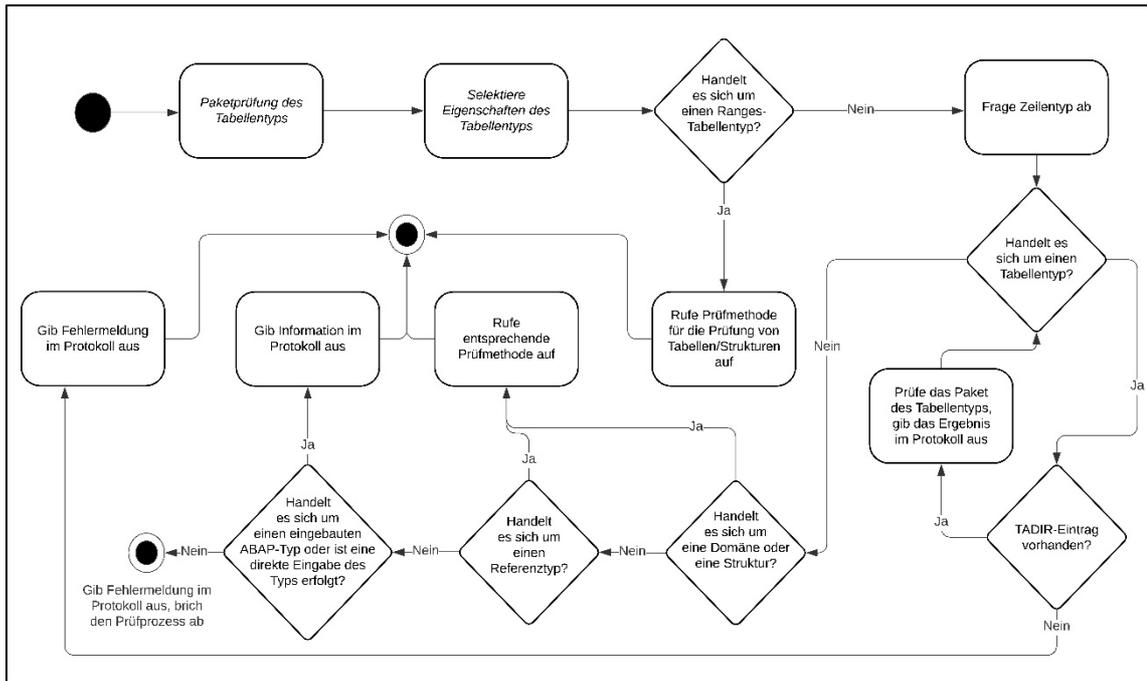
Darstellung 11: Beispielhaftes Ergebnis der Paketprüfung eines Tabellentyps (eigene Darstellung)

In Darstellung 11 ist eine durch die *CHECK_PACKAGES_OF_TABLE_TYPE*-Methode generierte Logausgabe dargestellt. Evident ist der Prüfprozess für den Tabellentyp */DEMATI/TT_SPOC01*. Der Darstellung ist zu entnehmen, dass das Paket, in dem der Tabellentyp gekapselt ist (*/DEMATI/DMON*), für den Export ausgewählt wurde. Zudem wurde die Struktur */DEMATI/SPOC01*, die die Zeilen des Tabellentyps beschreibt, samt ihrer Komponenten einer Prüfung unterzogen. Zur Prüfung der Struktur wurde die in Abschnitt 4.4 beschriebene Prüfmethode angewandt. Das Prüfergebnis weist darauf hin, dass alle benötigten Pakete im Dematic TB für den Export ausgewählt wurden.

4.3.1 Modultestergebnis

Die durchgeführten Modultests konnten fehlerfrei abgeschlossen werden.

4.3.2 Übersicht über den Prüfprozess eines Tabellentyps



Darstellung 12: Flussdiagramm des Prüfprozesses eines Tabellentyps (eigene Darstellung)

Der in Darstellung 12 abgebildete Prüfprozess eines Tabellentyps wird nachfolgend erläutert:

Zunächst wird die Paketprüfung des Tabellentyps vorgenommen. Hierfür wird die in Abschnitt 4.1.4 dargestellte Prüflogik mit den dort beschriebenen Anpassungen angewandt. Daraufhin folgt die Selektion der Eigenschaften des Tabellentyps. Auf Basis der Art des Tabellentyps (Ranges- oder allgemeiner Tabellentyp) wird über das weitere Prüfprozedere entschieden. Handelt es sich um einen Ranges-Tabellentyp, erfolgt dessen Paketprüfung durch die in Abschnitt 4.4 aufgezeigte Prüfmethode für die Prüfung von Tabellen/Strukturen. Andernfalls wird auf Grundlage des festgelegten Zeilentyps entschieden, ob und wie die Prüfung zu erfolgen hat.

Wird der Zeilentyp eines Tabellentyps durch einen weiteren Tabellentyp festgelegt, wird zunächst der *TADIR*-Eintrag des festlegenden Zeilentyps geprüft. Ist dieser nicht zu ermitteln, wird der Prüfprozess abgebrochen. Zudem wird eine Fehlermeldung im Protokoll vermerkt, um darauf hinzuweisen, dass die Pakete dieses Tabellentyps nicht geprüft werden konnten. Wurden ein *TADIR*-Eintrag und somit auch ein Paket für den Zeilentyp ermittelt, erfolgt die Prüfung der Paketzugehörigkeit dieses Zeilentyps. Das Ergebnis wird im Protokoll dokumentiert. Anschließend wird erneut überprüft, ob der Zeilentyp des geprüften Tabellentyps durch einen Tabellentyp festgelegt wird. Ist dies der Fall, wird die Schleife so lange erneut durchlaufen, bis der Zeilentyp nicht mehr durch einen Tabellentyp, sondern durch eine andere Datenstruktur (z. B. eine Struktur) beschrieben wird.

Wird der Zeilentyp des Tabellentyps durch eine andere Datenstruktur oder durch keinen weiteren Tabellentyp mehr beschrieben, wird auf Basis des Objekts, durch das der Zeilentyp festgelegt wird, entschieden, ob Protokollausgaben zu erfolgen haben bzw. ob weitere Prüfmethode aufgerufen werden müssen.

4.4 Prüfung von Tabellen und Strukturen

In diesem Modul werden die unter Abschnitt 3.3.1 genannten Prüffälle 3 und 4 implementiert. Hierfür werden zwei neue Methoden angelegt, die im Folgenden erklärt werden.

Mittels der *CHECK_PACKAGES_OF_STRUC_TABL*-Methode wird die Paketprüfung von Strukturen und Tabellen durchgeführt. Hierfür findet eine Iteration über die zu prüfenden Strukturen und/oder Tabellen statt. Anschließend erfolgt die Prüfung der Paketzugehörigkeit der jeweiligen Tabelle oder Struktur und die Prüfung der einzelnen Komponenten der zu untersuchenden Struktur bzw. derjenigen Struktur, die die Felder der zu untersuchenden Tabelle beschreibt. Mittels einer Selektion der *DD03L*-Tabelle, werden die Komponenten (einschließlich der Komponenten eventuell vorhandener Unterstrukturen) der zu prüfenden Datenstruktur selektiert.

Von besonderer Bedeutung für diesen Prüfprozess ist das Tabellenfeld *COMPTYPE*, in dem angegeben wird, um welchen Komponententyp es sich handelt. Insgesamt lassen sich vier Komponententypen unterscheiden, die für die Paketprüfung zu berücksichtigen sind. Da sich einige in ihrer Behandlung ähneln, werden diese zu drei Fällen zusammengefasst:

Fall 1:

Eine Komponente wurde mit Bezug auf ein Datenelement oder einen Tabellentyp definiert. Die Komponentenprüfung wird mittels der bereits vorhandenen *CHECK_PACKAGES_OF_DATA_ELEMENT*- (vgl. Abschnitt 4.2) bzw. *CHECK_PACKAGES_OF_TABLE_TYPE*-Prüfmethode (vgl. Abschnitt 4.3) durchgeführt.

Fall 2:

Der Typ der zu prüfenden Komponente ist ein eingebauter ABAP-Typ. Dieser wird einem ABAP-Programm von der ABAP-Laufzeitumgebung zur Verfügung gestellt. Aus diesem Grund muss ein solcher Typ nicht in ein Zielsystem transportiert werden. Es ist keine Prüfung erforderlich – dies wird im Protokoll vermerkt.

Fall 3:

Der Typ der zu prüfenden Komponente ist ein Referenztyp. Für die Handhabung von Referenztypen wird eine neue Methode angelegt.

Mithilfe der *CHECK_PACKAGE_OF_REFERENCE_TYP*-Methode erfolgt hierbei die Prüfung der Paketzugehörigkeit eines einzelnen Referenztyps. Insgesamt lassen sich

sieben unterschiedliche Komponenten unterscheiden, die referenziert werden können. Da sich einige Komponenten in ihrer Handhabung gleichen, werden sie zu drei Fallgruppen zusammengefasst:

Fall 3a:

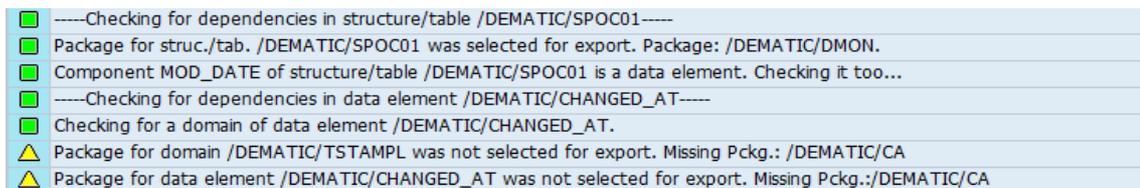
Es wird eine Klasse oder eine Schnittstelle referenziert. Bei diesem Fall wird lediglich die Klasse bzw. die Schnittstelle hinsichtlich der Paketzugehörigkeit geprüft. Anschließend wird eine Warnung bzw. eine Information über das Testergebnis im Protokoll ausgegeben. Eine weitere Paketprüfung, wie sie z. B. bei Klassenkomponenten denkbar wäre, ist vom Auftraggeber nicht gewünscht.

Fall 3b:

Ein Datenelement, ein Tabellentyp oder eine Tabelle bzw. eine Struktur wird referenziert. Die Prüfung der Paketzugehörigkeit erfolgt mittels der Prüfmethode `CHECK_PACKAGES_OF_DATA_ELEMENT` (vgl. Abschnitt 4.2), `CHECK_PACKAGES_OF_TABLE_TYPE` (vgl. Abschnitt 4.3) bzw. `CHECK_PACKAGES_OF_STRUC_TABL` (vgl. Abschnitt 4.4).

Fall 3c:

Bei der referenzierten Komponente handelt es sich um einen eingebauten ABAP-Typ. Wie in Fall 2 bereits beschrieben, wird eine Information im Protokoll ausgegeben.



| | |
|---|---|
| ■ | ----Checking for dependencies in structure/table /DEMATI/SPOC01---- |
| ■ | Package for struc./tab. /DEMATI/SPOC01 was selected for export. Package: /DEMATI/DMON. |
| ■ | Component MOD_DATE of structure/table /DEMATI/SPOC01 is a data element. Checking it too... |
| ■ | ----Checking for dependencies in data element /DEMATI/CHANGED_AT---- |
| ■ | Checking for a domain of data element /DEMATI/CHANGED_AT. |
| ▲ | Package for domain /DEMATI/TSTAMPL was not selected for export. Missing Pckg.: /DEMATI/CA |
| ▲ | Package for data element /DEMATI/CHANGED_AT was not selected for export. Missing Pckg.:/DEMATI/CA |

Darstellung 13: Beispielhaftes Ergebnis der Paketprüfung einer Struktur (eigene Darstellung)

Der Darstellung 13 ist eine durch die `CHECK_PACKAGES_OF_STRUC_TABL`-Methode generierte Protokollausgabe zu entnehmen. Es wird der Prüfvorgang für die Struktur `/DEMATI/SPOC01` dargestellt. Ersichtlich ist das Prüfergebnis der Struktur sowie der Komponente `MOD_DATE`, bei der es sich um ein Datenelement handelt. Dieses Element wurde mittels der in Abschnitt 4.2 beschriebenen Prüfmethode geprüft. In diesem Fall wurde weder das Paket der Struktur noch dasjenige des Datenelements oder dasjenige der dem Datenelement zugehörigen Domäne im Dematic TB für den Export ausgewählt. Auf diesen Umstand wird mittels gelber Warnungen hingewiesen.

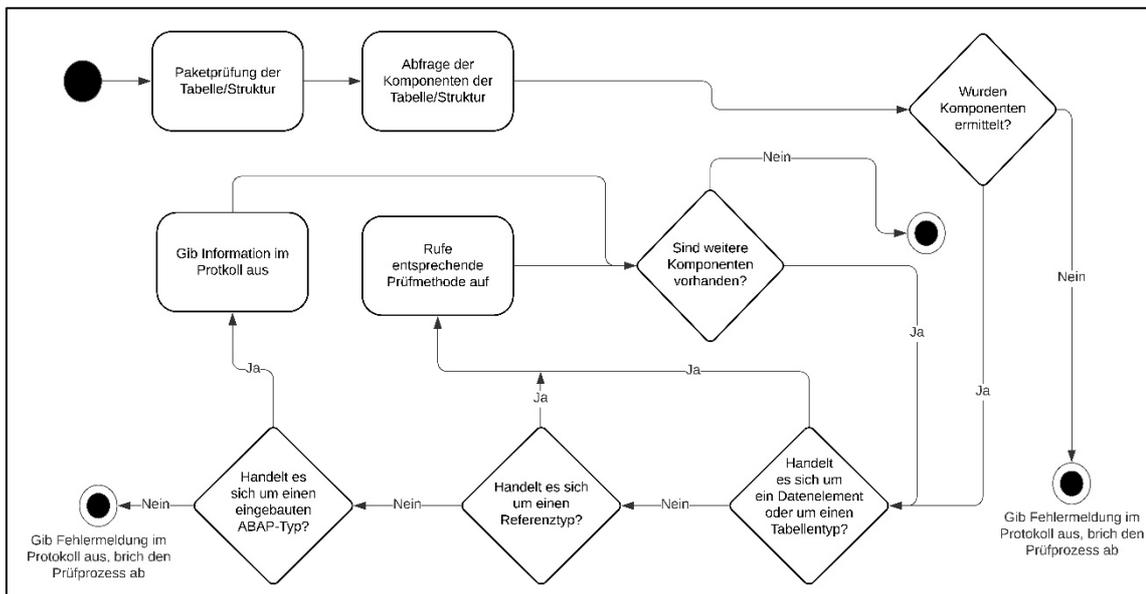
4.4.1 Modultestergebnis

Die Modultests für das vierte Modul waren umfangreich, jedoch hielt sich die Anzahl der gefundenen Fehler aufgrund des modularen Anwendungsaufbaus in Grenzen. Ein Fehler in der `CHECK_PACKAGES_OF_STRUC_TABL`-Methode führte dazu, dass das Programm abstürzte. Dieser Fehler lag immer dann vor, wenn zu einer Tabelle oder Struktur keine Felder ermittelt werden konnten (z. B. weil die Tabelle oder Struktur

gelöscht wurde). Dies hat zur Folge, dass auf ein nicht initialisiertes Feldsymbol⁵ zugegriffen wird, wodurch eine Ausnahme ausgelöst wird. Der Fehler wird wie folgt behoben: Der Prüfprozess wird nun abgebrochen, wenn keine Felder einer Tabelle bzw. einer Struktur aufgefunden werden. Anschließend wird eine Fehlermeldung im Protokoll ausgegeben. Außerdem wird die Ausnahmebehandlung in der gesamten Anwendung überarbeitet: Alle auftretenden Ausnahmen werden nun korrekt behandelt, um Abstürze des Programms zu verhindern.

Zahlreiche weitere Fehler betrafen u. a. doppelte Protokollausgaben und Tippfehler. Zudem lag eine inkorrekte Paketprüfung von Objekten vor. Die Fehler werden mit geringem Arbeitsaufwand behoben.

4.4.2 Übersicht über den Prüfprozess einer Tabelle bzw. einer Struktur



Darstellung 14: Flussdiagramm des Prüfprozesses einer Tabelle bzw. einer Struktur (eigene Darstellung)

In der Darstellung 14 ist das Prüfprozedere einer Tabelle bzw. einer Struktur abgebildet. Es wird ersichtlich, dass anfänglich die Paketprüfung der Tabelle bzw. der Struktur vorgenommen wird. Auch für diesen Prozess wird auf die in Abschnitt 4.1.4 beschriebene Prüflogik mit den für die Prüfung von Tabellen bzw. für Strukturen vorgenommenen Anpassungen zurückgegriffen. Darauf folgt die Ermittlung der Komponenten des zu prüfenden Objekts. Werden keine festgestellt, wird der Prüfprozess abgebrochen und im Protokoll wird eine Fehlermeldung ausgegeben. Werden Komponenten ermittelt, wird auf Basis des Komponententyps über das weitere Vorgehen entschieden. Handelt es sich bei der gesuchten Komponente z. B. um ein

⁵ Bei Feldsymbolen handelt es sich um Platzhalter für bestehende Datenobjekte bzw. für Teile von bestehenden Datenobjekten, denen zur Programmlaufzeit ein Speicherbereich zugewiesen werden kann [SA21m]. Ein Zugriff auf ein uninitialisiertes Feldsymbol löst eine Ausnahme aus.

Datenelement, einen Referenztyp oder einen Tabellentyp, wird die entsprechende Prüfmethode eines vorherigen Moduls aufgerufen, um eine solche Datenstruktur hinsichtlich ihrer Paketzugehörigkeit prüfen zu können. Im Falle eines eingebauten ABAP-Typs wird – ähnlich den Prüffällen anderer Module – eine Information im Protokoll ausgegeben, der zu entnehmen ist, dass keine weitere Prüfung erforderlich ist. Nach jedem Protokolleintrag bzw. jedem Aufrufen einer Prüfmethode wird kontrolliert, ob die Tabelle bzw. die Struktur weitere Komponenten hat. Ist dies der Fall, wird erneut auf Basis des Objekttyps entschieden, welches der oben genannten Prozedere angewandt wird. Andernfalls wurde die Tabelle bzw. die Struktur einer vollständigen Paketprüfung unterzogen.

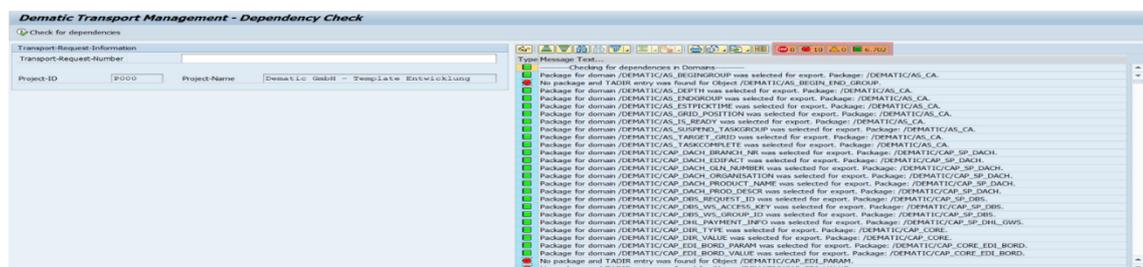
4.5 Finale Anwendungsarchitektur

Dieser Abschnitt beschreibt die Vervollständigung der Benutzerschnittstelle, die um in Abschnitt 4.5.1 beschriebene Änderungswünsche der Auftraggeber ergänzt wird.

4.5.1 Die Benutzerschnittstelle – Protokollausgabe

Die in Abschnitt 3.3.2 dargestellte Protokollausgabe wird durch eine Custom Control realisiert. Um in dieser das entsprechende Protokoll darstellen zu können, wird die *CHECK_ALL_OBJECTS*-Methode erweitert. Mittels dieser Methode wird das Protokoll-Handle⁶ des zuvor beschriebenen Protokolls zurückgegeben.

Außerdem wird das Programm */DEMATI/TMS_CHECK* um ein Unterprogramm ergänzt: *SHOW_LOG* dient dem Laden des Protokolls und dem Anzeigen der Protokollausgaben in der Custom Control. Hierfür wird mittels des übergebenen Protokoll-Handles der Header des Protokolls ermittelt, mit dem wiederum das Protokoll in den Hauptspeicher geladen wird. Darauf aufbauend wird mithilfe eines Funktionsbausteins⁷, der von der SAP bereitgestellt wird, die Custom Control mit dem Protokoll befüllt. Da Letzteres bereits eine von der SAP implementierte Filterfunktionalität beinhaltet (nachfolgend durch ein kräftiges Rot hervorgehoben), wird auf die im Konzept ersichtliche Filterfunktion verzichtet.



Darstellung 15: Benutzerschnittstelle mit Protokollausgabe (eigene Darstellung)

⁶ Das Protokoll-Handle ist ein global eindeutiger Bezeichner, mit dem ein Protokoll eindeutig identifiziert wird.

⁷ Funktionsbausteine sind programmübergreifend wiederverwendbare Prozeduren [SA21n].

Nachdem die Benutzerschnittstelle vollständig umgesetzt wurde, wurde sie den Auftraggebern präsentiert. Diese äußerten hierbei einige Verbesserungswünsche, deren Implementierung nachfolgend behandelt wird.

Die in Abschnitt 3.1 beschriebenen funktionalen Anforderungen werden – auf Wunsch der Auftraggeber – um folgende ergänzt:

- Das Dynpro soll den Eigentümer und die Kurzbeschreibung des eingegebenen Transportauftrags anzeigen.
- Das Dynpro soll das Protokollobjekt⁸ des generierten Protokolls anzeigen.
- Das Dynpro soll es dem Anwender ermöglichen, Informationsausgaben im generierten Protokoll zu aktivieren bzw. zu deaktivieren.
- Anhand des Dynpros soll der Anwender die Speicherdauer des Protokolls festlegen können.
- Mit dem Dynpro soll der Anwender bestimmen können, welche Informationen (z. B. nur Warnungen) im Protokoll gespeichert werden sollen.

Da die gewünschten Änderungen hauptsächlich das bereits implementierte Dynpro betreffen, wird auf Basis der Änderungswünsche ein überarbeiteter visueller Prototyp vorgestellt:

Transport-Request-Number Trans.-Requ.-Owner
Trans.-Requ.-Name
Project-ID Project-Name
Log: Object name Log: Subobject
Log Active Global No Info-Messages Validity of Log (DAYS)

LOG

Darstellung 16: Visueller Prototyp der finalen Benutzerschnittstelle (eigene Darstellung)

Die neuen Felder, deren Beschriftungen durch die Dematic GmbH vorgegeben wurden, sollen nachfolgend vorgestellt werden:

Trans.-Requ.-Owner und *Trans.-Requ.-Name* dienen der Ausgabe des Eigentümers und der Kurzbeschreibung des eingegebenen Transportauftrags. Beide werden durch eine Selektion der *E070V*-Tabelle ermittelt. Mittels der Ausgabefelder *Log: Object name* und

⁸ Jedes Protokoll wird zu einem Objekt und ggf. zu einem Unterobjekt, das zu diesem Objekt gehört, erfasst. Mittels dieser Objekte kann ein Protokoll effizient wiedergefunden werden.

Log: Subobject wird der Benutzer über das Protokollobjekt bzw. über das Protokollsubobjekt informiert. Da sich diese für das entwickelte Programm nicht ändern, werden beide Felder mit konstanten Werten belegt. Mit dem Eingabefeld *Log Active Global* kann der Anwender festlegen, welche Informationen im Protokoll gespeichert werden sollen. Hierfür kann der Benutzer zwischen verschiedenen Optionen (z. B. nur sehr wichtige Informationen speichern oder nur wichtige Informationen speichern) wählen. Das Ankreuzfeld *No-Info-Messages* und das Eingabefeld *Validity of Log (DAYS)* ermöglichen es dem Anwender, Informationsausgaben im generierten Protokoll zu aktivieren oder zu deaktivieren bzw. eine minimale Speicherdauer für das Protokoll anzugeben.

Der Darstellung 17 ist die gemäß den Änderungswünschen angepasste Benutzerschnittstelle zu entnehmen:

Darstellung 17: Vervollständigte Benutzerschnittstelle (eigene Darstellung)

4.5.2 Testergebnis

Anhand der durchgeführten Tests konnte folgender Fehler in der Benutzerschnittstelle ermittelt werden:

Die Eingabe einer ungültigen Transportauftragsnummer führte dazu, dass keine weiteren Benutzereingaben mehr möglich waren. Dieses Problem kann durch eine neu implementierte Eingabebehandlung behoben werden: Die Gültigkeit von Nutzereingaben wird nun überprüft, bevor weitere Programmlogik ausgeführt wird.

5 Schluss

In diesem Kapitel werden zunächst der Planungs- und Entwicklungsprozess des Programms zusammengefasst. In einem anschließenden Fazit werden die erreichten Ergebnisse vorgestellt. Es folgt ein Ausblick über mögliche zukünftige Erweiterungen des umgesetzten Programms.

5.1 Zusammenfassung

Mit dieser Arbeit wird das Ziel verfolgt, ein Programm zu entwickeln, mit dem SAP-Softwaretransportaufträge auf Entwicklungsobjekte hin untersucht werden, die Objekte referenzieren, deren Pakete nicht Teil der im Dematic TB ausgewählten Paketstruktur sind. Durch die Ausgabe solcher referenzierter Objekte in einem Protokoll wird der Projektleitung die Möglichkeit gegeben, den auszuliefernden Projektumfang (vgl. Abschnitt 2.3) anzupassen, um Nachtransporte fehlender Objekte zu vermeiden.

Um das Thema dieser Arbeit umzusetzen, werden zunächst in einer Projektbesprechung mit dem Auftraggeber die der Anwendung zugrunde liegenden Anforderungen erhoben (vgl. Abschnitt 3.1). Hierbei wird zugleich eine Einschränkung des Prüfalgorithmus auf ausgewählte Objekttypen, die bei der Dematic GmbH für einen Großteil aller Nachtransporte verantwortlich sind, vorgenommen.

Um den Softwareentwicklungsprozess übersichtlich und beherrschbar zu gestalten, wird, aufbauend auf den ermittelten Anforderungen, ein Vorgehensmodell zur Softwareentwicklung ausgewählt (vgl. Abschnitt 3.2). Der Autor dieser Arbeit entscheidet sich für ein prototypisches Vorgehensmodell. Dieses hat zum Vorteil, dass mögliche unerkannte Anforderungen früh festgestellt werden können und bereits in frühen Entwicklungsphasen etwaige Probleme in den dem Prototyp zugrunde liegenden Konzepten erkannt werden können. Nach einer Analyse ausgewählter Prototyping-Varianten wird für den Entwicklungsprozess der Ansatz des evolutionären Prototyping ausgewählt, um zügig erste Ergebnisse erzielen zu können und auf eventuelle Änderungswünsche vonseiten des Auftraggebers schnell reagieren zu können.

Nach der Festlegung auf ein Vorgehensmodell, werden Konzepte für die Umsetzung der Anwendung erarbeitet (vgl. Abschnitt 3.3). Auf Basis der ermittelten Anforderungen werden ein Prüf- und ein Testkonzept entworfen. Ebenso wird eine Benutzerschnittstelle konzipiert.

Die getroffenen Designentscheidungen dienen als Grundlage für die inkrementelle Umsetzung des Programms (vgl. Kapitel 4). Der Fertigstellung eines jeden Inkrements folgen umfangreiche Modultests, um mögliche Fehler frühzeitig erkennen zu können. Zudem wird nach jedem Inkrement Rücksprache mit dem Auftraggeber gehalten, um eventuelle Änderungswünsche schnell erkennen und umsetzen zu können.

5.2 Fazit

Durch einen ausgiebigen Planungsprozess und ständigen Kontakt zur Dematic GmbH konnte das zu entwickelnde Programm ohne größere Probleme umgesetzt werden. Die entwickelte Software zeichnet sich durch eine intuitiv zu bedienende Benutzerschnittstelle aus, die der übersichtlichen Darstellung aller relevanten Informationen über den zu prüfenden Transportauftrag und des Ergebnisses der Prüfung dient (vgl. Abschnitt 4.5.1). Zudem wurde durch projektbegleitende Modultests die Zuverlässigkeit des implementierten Prüfalgorithmus sichergestellt. Die von der Paketprüfung ermittelten Objekte werden zusammen mit dem Prüfprozess übersichtlich im Protokoll dargestellt. Zudem wurde bereits im Planungsprozess die Erweiterbarkeit der Anwendung sichergestellt, da die Entwicklung eines in aufeinander aufbauenden Modulen aufgeteilten Prozesses der Paketprüfung stattfand (vgl. Abschnitt 3.3.1). Die Anforderungen, die dem entwickelten Programm zugrunde lagen (vgl. Abschnitt 3.1), konnten einschließlich der Änderungswünsche des Auftraggebers (vgl. Abschnitt 4.5.1) vollumfänglich erfüllt werden.

Aufgrund der Einschränkung der Paketprüfung auf ausgewählte Datenstrukturen (vgl. Abschnitt 3.1) und anhand einer abschließenden Vorstellung des Programms vor dem Führungspersonal der Dematic GmbH konnten signifikante Feststellungen gesammelt werden, welche der Weiterentwicklung der implementierten Anwendung dienen könnten. Diese werden nachfolgend aufgeführt.

5.3 Ausblick

Mit dem bisherigen Prüfalgorithmus werden ausschließlich einem Transportauftrag zugeordnete globale Datenstrukturen geprüft. Lokale Datendeklarationen, wie sie beispielsweise in Klassen auftreten, werden von dem Prüfprozedere nicht erfasst. Wurde z. B. in einer Klasse ein Objekt mit Bezug auf ein globales Datenelement lokal definiert, so kann es, wenn ein Export dieser Klasse stattfindet, vorkommen, dass das referenzierte Datenelement nicht mit in das Zielsystem exportiert wird. Dieser Umstand würde einen Nachtransport des Datenelements erforderlich machen.

Eine denkbare Erweiterung der bisherigen Implementierung wäre die Realisierung eines Algorithmus für das effiziente Auffinden von lokal definierten Objekten. Hierfür könnte z. B. ein endlicher Automat umgesetzt werden, mit dem lokal definierte Entwicklungsobjekte, die globale Datenstrukturen referenzieren, erkannt werden und anschließend den entsprechenden Prüfmethode übergeben werden. Zudem könnte ein Algorithmus implementiert werden, der automatisiert die als fehlend ermittelten Pakete im Dematic TB auswählt und so die Paketauswahl autonom vervollständigt. Denkbar wären auch Erweiterungen, mit denen nicht nur Domänen, Datenelemente, Strukturen,

Tabellen und Tabellentypen untersucht werden können, sondern auch andere Entwicklungsobjekte (z. B. Klassen), die Objekte referenzieren können, die Paketen zugeordnet sein können, die nicht als exportrelevant eingestuft wurden. Auch eine Ausweitung des Prüfprozesses auf Datenelemente, die der Beschreibung von Referenztypen dienen, wäre vorstellbar.

Literaturverzeichnis

- [AI18] Alexander Kreutner: Dematic TMS-Manager, 2018. (interner Vortrag)
- [Du21] Duden: Projekt. <https://www.duden.de/rechtschreibung/Projekt>, Stand: 07.09.2021.
- [Eb21] Eberstein, R.: Transportauftrag – SAP-Wiki. <https://www.berater-wiki.de/Transportauftrag>, Stand: 06.09.2021.
- [Ja09] Jansen, M.: Programmierung von SAP-Systemen: Eine Einführung in ABAP. Books on Demand, Norderstedt, 2009.
- [SA21a] SAP Datasheet: SAP ABAP Table DD03L (Tabellenfelder). <https://www.sapdatasheet.org/abap/tabl/dd03l.html>, Stand: 13.12.2021.
- [SA21b] SAP Datasheet: SAP ABAP Table DD40L (Tabellentypen (im DD definierte interne Tabellen)). <https://www.sapdatasheet.org/abap/tabl/dd40l.html>, Stand: 13.12.2021.
- [SA21c] SAP Datasheet: SAP ABAP Table E070 (Transportsystem: Header von Aufträgen/Aufgaben). <https://www.sapdatasheet.org/abap/tabl/e070.html>, Stand: 13.12.2021.
- [SA21d] SAP Datasheet: SAP ABAP Table E070V (Generierte Tabelle zum View E070V). <https://www.sapdatasheet.org/abap/tabl/e070v.html>, Stand: 13.12.2021.
- [SA21e] SAP Datasheet: SAP ABAP Table E071 (Transportsystem: Objekt-Einträge von Aufträgen/Aufgaben). <https://www.sapdatasheet.org/abap/tabl/e071.html>, Stand: 13.12.2021.
- [SA21f] SAP Datasheet: SAP ABAP Table TADIR (Katalog der Repository-Objekte). <https://www.sapdatasheet.org/abap/tabl/tadir.html>, Stand: 13.12.2021.
- [SA21g] SAP SE: ABAP - Dictionary. https://help.sap.com/doc/abapdocu_752_index_htm/7.52/de-de/abenabap_dictionary.htm, Stand: 10.12.2021.
- [SA21h] SAP SE: Custom Controls. https://help.sap.com/doc/saphelp_nw73/7.3.16/de-DE/4a/4d572f2fb01d0fe10000000a42189c/frameset.htm, Stand: 09.10.2021.

- [SA21i] SAP SE: Datenbanktabellen.
https://help.sap.com/doc/abapdocu_750_index_htm/7.50/de-de/abenddic_database_tables.htm, Stand: 18.09.2021.
- [SA21j] SAP SE: Datenelemente.
https://help.sap.com/doc/abapdocu_750_index_htm/7.50/de-de/abenddic_data_elements.htm, Stand: 14.10.2021.
- [SA21k] SAP SE: Domänen.
https://help.sap.com/doc/abapdocu_752_index_htm/7.52/de-de/abenddic_domains.htm, Stand: 13.02.2022.
- [SA21l] SAP SE: Datentypen.
https://help.sap.com/doc/abapdocu_752_index_htm/7.52/de-de/abendata_types.htm, Stand: 13.02.2022.
- [SA21m] SAP SE: Feldsymbole.
https://help.sap.com/doc/abapdocu_752_index_htm/7.52/de-DE/abenabap_field_symbols.htm, Stand: 03.10.2021.
- [SA21n] SAP SE: Funktionsbausteine.
https://help.sap.com/doc/abapdocu_751_index_htm/7.51/de-DE/abenabap_functions.htm, Stand: 14.10.2021.
- [SA21o] SAP SE: Interne Tabellen - Übersicht.
https://help.sap.com/doc/abapdocu_752_index_htm/7.52/de-DE/abenitab_oview.htm, Stand: 14.10.2021.
- [SA21p] SAP SE: Open SQL - Übersicht.
https://help.sap.com/doc/abapdocu_751_index_htm/7.51/de-DE/abenopen_sql_oview.htm, Stand: 05.10.2021.
- [SA21q] SAP SE: Paket.
https://help.sap.com/doc/abapdocu_751_index_htm/7.51/de-DE/abenpackage_glosry.htm, Stand: 10.09.2021.
- [SA21r] SAP SE: Protokolle anzeigen.
https://help.sap.com/doc/saphelp_nw75/7.5.5/de-DE/4e/23ac220771417fe10000000a15822b/frameset.htm, Stand: 11.12.2021.
- [SA21s] SAP SE: Selektionstabelle.
https://help.sap.com/doc/abapdocu_751_index_htm/7.51/de-DE/abenselection_table_glosry.htm, Stand: 30.09.2021.

- [SA21t] SAP SE: Strukturen.
https://help.sap.com/doc/abapdocu_752_index_htm/7.52/de-de/abendata_objects_structure.htm, Stand: 03.12.2021.
- [SA21u] SAP SE: Tabellentypen.
https://help.sap.com/doc/abapdocu_751_index_htm/7.51/DE/abenddic_table_types.htm, Stand: 11.12.2021.
- [Wi11] Wiczorrek, H. W.: Management von IT-Projekten: Von der Planung zur Realisierung. 4. Aufl. Springer, Berlin, 2011.

Abkürzungsverzeichnis

| | |
|--------------------|--|
| Dematic TB | <i>Dematic Transport Builder</i> |
| Dematic TMS | <i>Dematic Project And Transport Management System</i> |
| Dynpro | <i>Dynamisches Programm</i> |
| Elementartyp | <i>Eingebauter elementarer Datentyp</i> |