Teilautomatisierung der Testspezifikation einer Blasmaschine der Krones AG

Phong Tran BSc.
Professor Dr.-Ing. Frank Herrmann
Ostbayerische Technische Hochschule Regensburg
Innovationszentrum für Produktionslogistik und Fabrikplanung (IPF)
E-Mail: Frank.Herrmann@OTH-Regensburg.de

SCHLÜSSELWÖRTER

Digitalisierung, Prozessverbesserung, SAP Transaktion.

ABSTRACT

In der Krones AG werden alle Blasmaschinen eines neuen Kundenauftrags – sowohl als Teil einer Komplettanalage als auch einzeln – Vortests unterzogen, um sicherzustellen, dass die Maschine die kundenspezifischen Anforderungen an die speziellen Konfigurationen der Flaschenform erfüllt. Diese Prüfungen erfordern Testmaterialien, bei denen es sich im Fall der Blasmaschinen um Behälter – auch Preforms genannt – handelt, die als Ausgangsprodukt für die Herstellung der eigentlichen Flaschen verwendet werden. Aus diesem Grund müssen mit jedem neuen Kundenauftrag, der eine Blasmaschine beinhaltet, Testmaterialien in ausreichender Anzahl angefordert werden. Diese werden firmenintern produziert und müssen je nach Maschinenkonfiguration in der benötigten Anzahl bereitgestellt werden, um ausführliche Tests durchzuführen. Die Bestellungen erfolgen bisher stets mittels Formulare in Form von Word-Dokumenten, auch Flaschenanforderungen genannt, die manuell vom Vertrieb ausgefüllt werden. Dieser Vorgang erfordert viel Zeit und Ressourcen und soll aus diesem Grund automatisierter ablaufen. Ziel dieses Projekts ist eine Konzeption und Umsetzung einer neuen Funktion, die den alten Prozess ablöst und durch eine effizientere Lösung ersetzt. Hierbei soll zunächst die Funktionalität als eigenständiges Programm entwickelt werden. Benötigt wird eine Konzeption und Implementierung einer geeigneten graphischen Oberfläche sowie ein Programm, das automatisch die benötigten Daten zusammenfasst und diese in übersichtlicher Form darstellt.

Die Krones AG

Nach Angaben der Krones AG ist das Unternehmen Weltmarktführer in den Bereichen Maschinen- sowie Anlagenbau für Getränkeabfüll- und Verpackungsmaschinen. Weltweit beschäftigt das Unternehmen über 14.000 Mitarbeiter, wovon rund 9.800 in den fünf deutschen Niederlassungen, in Neutraubling, Freising, Rosenheim, Nittenau und Flensburg tätig sind, sowie circa 4.500 weitere Mitarbeiter, die weltweit tätig sind. Über 80 Vertriebs- und Serviceniederlassungen gehören zum Krones Konzern. Die Produktion erfolgt weitestgehend in Deutschland. 90% der im Inland produzierten Anlagen werden exportiert. Der Gesamtumsatz betrug im Geschäftsjahr 2015 3.174 Mrd. Euro (siehe (Krones AG, 2016)).

Geschäftsprozess-Analyse

Flaschenanforderungen an Kunden

Die Krones AG benutzt – um die für Kunden gebauten Maschinen zu testen – originale Testmaterialien des jeweiligen Kunden. Bei diesen Materialien kann es sich um Flaschen, Etiketten oder auch Füllgut wie etwa Bier, Softgetränke, Soßen oder dergleichen handeln.
Wenn Kunden eine Blasmaschine erwerben – ob Teil einer Komplettanalage (siehe Abbildung 1: Beispiel einer Komplettanlage mit Blasmaschine) oder einzeln, so werden dafür ebenfalls Testmaterialien benötigt, um sicherzustellen, dass diese Maschine die Flaschen des Kunden maßgetreu produziert. Bei den Materialien handelt es sich um Behälter – auch Performs genannt – die von Blasmaschinen mithilfe von Heißluft in die gewünschte Flaschenform geblasen werden. Diese werden zunächst durch die Abteilung Kundenobjekte/Testmaterial erfasst. Die Ermittlung der exakten Anzahl erfolgt dann durch den Vertrieb, welcher jede Auftragsposition des Kundenauftrags durchgeht und die benötigten Preforms zusammenaddiert.

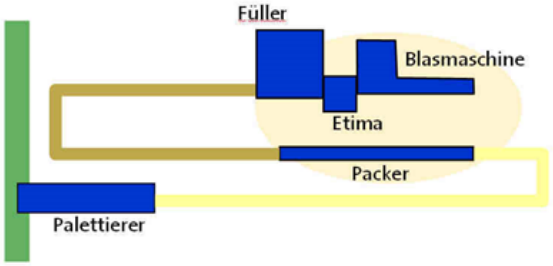

Abbildung 1: Beispiel einer Komplettanlage mit Blasmaschine

Diese Daten werden in ein Word-Dokument, welches als Flaschenanforderung bezeichnet wird, geschrieben. Nach Überprüfung der Informationen auf Korrektheit wird das ausgefüllte Formular ausgedruckt und an den nächsten Sachbearbeiter weitergeschickt wird.
Dieses Dokument muss für jeden neuen Vertriebsbeleg, der eine Blasmaschine beinhaltet, neu angelegt und versendet werden. Täglich werden im Schnitt 4 Flaschenanforderungen – im Monat ungefähr 88 – benötigt, die für

das erstmalige Ausfüllen schätzungsweise 30 Minuten in Anspruch nehmen. Einmalige nachträgliche Änderungen treten in ungefähr 30% aller Anforderungen auf und haben eine Bearbeitungszeit von zusätzlich ungefähr 10 Minuten. Des Weiteren sind in weiteren 30% zwei Änderungen erforderlich, die folglich 20 weitere Minuten beanspruchen. Dadurch ergibt sich folgende Rechnung:

Erstausfüllung	88 Anträge × 30 min = 44,0 h
1. Nachbesserung	88 Anträge × 0,3 × 10 min = 4,4 h
2. Nachbesserung	88 Anträge × 0,3 × 20 min = 8,8 h
Gesamt	**57,2 h**

Somit entsteht ein monatlicher Zeitaufwand von ungefähr 58 Arbeitsstunden.
Aus diesem Prozess (siehe Abbildung 2: Prozessablauf bei Flaschenanforderungen durch ein Ablaufdiagramm, das sich, wie auch die folgenden, an denen in der Literatur wie (Gadatsch, 2015) orientiert) werden folgende Probleme ersichtlich:

- Das Ausstellen des Dokuments stellt einen erheblichen Zeitaufwand für den Vertrieb dar, da Eingaben aktuell manuell ermittelt und eingegeben werden müssen.
- Bei jeder Änderung am Kundenauftrag muss der Erstellungsprozess wegen der manuellen Ermittlung der erforderlichen Informationen zu ca. 80% erneut durchgeführt werden.
- Die Flaschenanforderungen werden in Papierform verwaltet. Gerade bei häufigen Nachbesserungen erfordert dies einen hohen manuellen Aufwand.

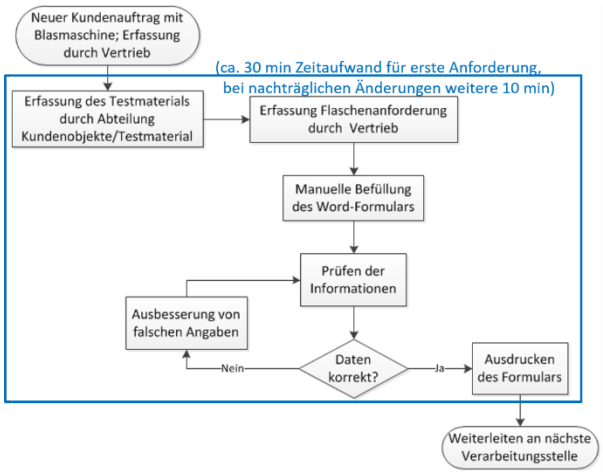

Abbildung 2: Prozessablauf bei Flaschenanforderungen

Der Prozess der Dokumentenerstellung soll automatisiert ablaufen. Dies soll in SAP durch eine neue Transaktion verwirklicht werden, wodurch sich folgender Ablauf ergibt:

- Bei einem Kundenauftrag mit einer Blasmaschine erhält der Vertrieb über die Transaktion alle benötigten Behälter mit ihrer jeweiligen benötigten Flaschenanzahl.
- Manuell überprüft der Vertrieb die Informationen und bessert diese in Sonderfällen aus.
- Das Formular wird auf „per Knopfdruck“ automatisch befüllt und per E-Mail versendet.

Des Weiteren soll das Auslesen, die Pfege und die Speicherung aller relevanten Daten in Datenbanktabellen im SAP System erfolgen.
Die Bearbeitungszeit für die Erstbefüllung des Formulars würde sich damit schätzungsweise auf 10 Minuten reduzieren und weitere Anpassungen auf jeweils 5 weitere Minuten. Daraus ergibt sich folgende Rechnung:

Erstausfüllung	88 Anträge × 10 min = 14,6 h
1. Nachbesserung	88 Anträge × 0,3 × 5 min = 2,2 h
2. Nachbesserung	88 Anträge × 0,3 × 10 min = 4,4 h
Gesamt	**21,2 h**

Somit entsteht eine monatliche Gesamtbearbeitungszeit von ungefähr 22 Stunden, welche im Vergleich zum derzeitigen Prozess eine Einsparung von 36 Stunden, also ungefähr 68% bedeutet.
Folgender Mehrwert würde sich durch die Einführung der neuen Transaktion ergeben:

- Enorme Zeitersparnis für den Vertrieb durch automatische Erstellung der Flaschenanforderungen.
- Relevante Informationen zum Einrichten der Maschinen bezüglich Behälter und Formulare sind komplett in elektronischer Form im System abgelegt und können dort auch kontrolliert gepflegt werden. Dadurch wird die Übersichtlichkeit beibehalten.
- Bei Änderungen des Kundenauftrags können die Daten der vorherigen Version der Flaschenanforderung abgerufen und ergänzt werden, sodass nur ein Teilprozess noch einmal durchlaufen werden muss.
- Eine im System integrierte Anwendung für Flaschenanforderungen kann auch zukünftig in anderen weltweiten Niederlassungen verwendet werden. Somit ist es möglich, diesen Prozess global einzuführen.
- Umweltschonend, da alles digital abläuft und kein Papier gedruckt werden muss.

Gegenstand dieses Projekts ist somit die Entwicklung, Implementierung und Test eines neuen Programms, das später durch einen Transaktionscode aufgerufen werden kann.

Konzeptionsphase

Um das Design und die Funktionen des Programms bestimmen zu können, muss zuerst festgelegt werden, wie der Prozess mit der Verwendung der neuen Software aussehen soll. Zur Prozessoptimierung sollen die Schritte der manuellen Befüllung und das Ausdrucken des Formulars automatisch durch das Programm übernommen werden. Lediglich die Prüfung der Daten und gegebenfalls eine Ausbesserung bei speziellen Fällen sollen manuell durchgeführt werden (siehe Abbildung 3: Konzept eines optimierten Prozessablaufs).

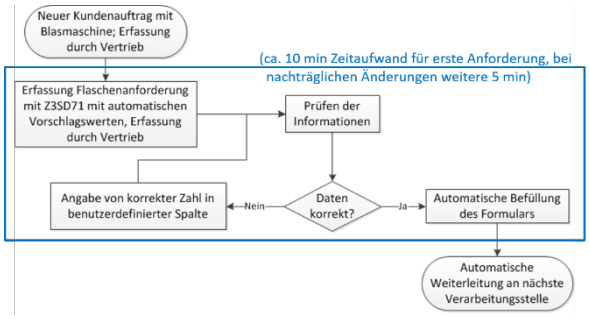

Abbildung 3: Konzept eines optimierten Prozessablaufs

Damit die neue Transaktion alle vorher genannten Vorteile vorzeigt, müssen folgende Funktionen integriert werden:

- Lesen und Zusammenfassen von Behälterdaten und den zugehörigen Preforms.
- Bereitstellung von Eingabefeldern für Benutzereingaben.
- Automatisches Auffüllen und Versenden des Formulars.
- Abspeichern von Benutzereingaben.

Daraus ergeben sich folgende funktionale Anforderungen (siehe auch Abbildung 4: Use-Case Diagramm zu den funktionalen Anforderungen, die in der Literatur üblichen Art und Weise dargestellt werden, s. z.B. (Oesterreich, 2009) oder (Balzert, 2011):

- Möglichkeit für Anwender, eine Verkaufsbelegnummer einzugeben.
- Übersichtliche Darstellung der Ergebnisse.
- Automatisches Befüllen des Word-Dokuments.
- Eingabemöglichkeiten für Anwender (Flaschenanzahl, Freitext).
- Abspeicherung von Benutzereingaben.

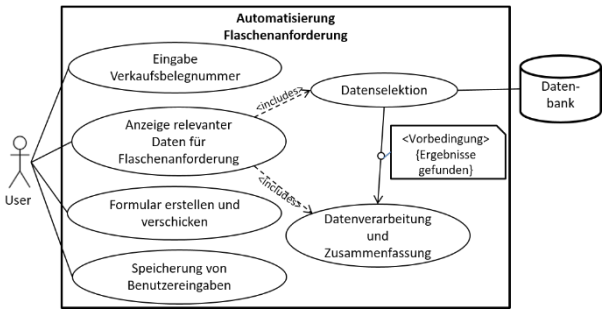

Abbildung 4: Use-Case Diagramm zu den funktionalen Anforderungen

Hierbei sind Korrektheit der Daten und angemessene Performance beim Lesen der Daten von Wichtigkeit. Die Verknüpfung von Vertriebsbeleg, Position, Equipment, Ausstattung und Kundenobjekt muss fehlerfrei ablaufen, sodass dem Anwender keine falschen Daten übergeben werden und das Dokument für die Flaschenanforderung nicht falsch befüllt wird.

Benötigte Klassen

Da die neue Funktion Teil eines größeren Pakets ist, sind bereits Klassen vorhanden, die für die Datenbeschaffung verwendet werden können. Die Klasse *ZCL_SDCC_DATA* liest relevante Daten anhand eines Vertriebsbelegs aus und fasst diese in Manager-Objekten zusammen, die in Equipments, Ausstattungen und Kundenobjekten aufgeteilt sind. Diese Objekte sind alle miteinander verbunden, da Kundenobjekte Teil von Ausstattungen sind, die wiederum zu bestimmten Equipments gehören (siehe Abbildung 5: Beziehung zwischen Equipment, Ausstattung und Kundenobjekt).

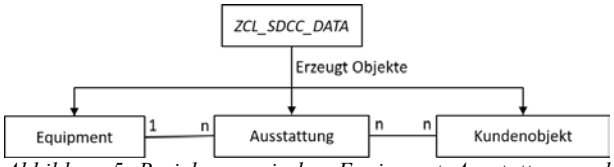

Abbildung 5: Beziehung zwischen Equipment, Ausstattung und Kundenobjekt

Das wichtigste Objekt ist in diesem Fall das der Klasse *ZCL_SDCC_EQ_MANAGER*, da anhand von diesem alle Equipments (*ZCL_SDCC_EQ_MANAGER*), Ausstattungen (*ZCL_SDCC_AST_SI*) und Kundenobjekte (*ZCL_SDCC_KO*) sowie deren Beziehungen ermittelt werden können (siehe Tabelle 1: Benötigte Klassen).

Klasse	Nutzen
ZCL_SDCC_DATA	Auslesen aller benötigten Daten zu einem Vertriebsbeleg
ZCL_SDCC_EQ_MANAGER	Liste mit Equipments
ZCL_SDCC_EQ	
ZCL_SDCC_AST_SI	Liste mit Ausstattungen
ZCL_SDCC_KO	Liste mit Kundenobjekten
	Informationen zu Kundenobjekten

Tabelle 1: Benötigte Klassen

Begriffe der ABAP Programmierung

Dynpro

Graphische Oberflächen, die der Interaktion zwischen dem SAP Programm und dem Benutzer dienen, werden in SAP Dynpros genannt. Mehrere Dynpros zusammen bilden Dynprofolgen, die laut (Keller & Krüger, 2006) über Dialogtransaktionen oder aus ABAP-Programmen aufgerufen werden können. Diese werden zur Kommunikation mit dem Anwender verwendet, um benutzerdefinierte Eingaben zu ermöglichen und das Ausgabeformat flexibler gestalten zu können. Ein Dynpro kann beliebig viele Dynpro-Felder beinhalten – wie zum Beispiel Ein- und Ausgabefelder oder Drucktasten – die im Programm über einen eindeutigen Namen angesprochen werden können. Vor Anzeige des Bildschirmbildes erfolgt das Ereignis PBO (Process before output), das dazu verwendet werden kann, Voreinstellungen zu bestimmen. Gleiches gilt für das Ereignis PAI (Process after input), das nach Auslösen einer Aktion aufgerufen wird und für die Verarbeitung von Benutzereingaben nützlich ist. Um ein

Dynpro in Unterbereiche aufzuteilen, werden Container verwendet, welche im Folgenden erläutert werden.

Docking- und Splitter-Container
Die in SAP genannten Container werden in ABAP zur Aufteilung der Benutzeroberfläche verwendet, um mehrere SAP List Viewer Objekte auf einer Oberfläche anzeigen zu können. Die Anzahl der Container ist nicht beschränkt, sodass die Oberfläche in beliebig viele kleinere Bereiche aufgeteilt werden kann.
Docking Container werden benutzt, um die Ausrichtung bestimmter Bereiche im Dynpro zu bestimmen. Dieser passt sich immer der individuellen Bildschirmauflösung an, sodass jeder Anwender die Benutzeroberfläche optimal ausnutzen kann. Beim Docking Container werden dessen Bildschirmbereiche an die Ränder von Dynpros angehängt. Dieser Bereich kann an alle Seiten der Ausgabe gebunden werden (oben, unten, links und rechts). Die Ausgabe wird dabei zu Gunsten des Docking Containers in der Größe angepasst.
Zur Teilung von Containern in Untercontainer werden die Splitter Container verwendet. Wird ein Splitter Container angelegt, wird die Ausgabe in einem bestimmten Verhältnis – welches angegeben werden muss – geteilt. Die Oberfläche wird dabei immer in zwei Teile gespalten. Mit dieser Funktionalität kann man je nach Belieben diverse Teilungen der Ausgabe vornehmen. Allerdings kann ein Splitter Container laut (Keller & Krüger, 2006) den Bereich horizontal und vertikal in maximal 16x16 Unterbereiche aufteilen. Für mehrere Teilungen wird ein bereits vorhandener Container nochmals aufgeteilt. Im Gegensatz zum Docking Container muss beim Instanzieren eines Splitter Containers ein bereits vorhandener Container als Eingabeparameter übergeben werden.

SAP List Viewer (ALV Grid, ALV Tree)
Der SAP List Viewer wird zur übersichtlichen und strukturierten Darstellung von Daten verwendet. Hierbei werden laut (Keller & Krüger, 2006) bereits bestimmte Funktionen standardmäßig implementiert, wie zum Beispiel:

- Druckvorschau.
- Sortieren.
- Filtern.
- Summieren.

Mit Hilfe des List Viewers können folgende beide Darstellungsvarianten umgesetzt werden. Sie basieren auf balancierten binären Suchbäumen, die das AVL-Kriterium erfüllen, welches von Adelson-Velsky und Landis 1962 eingeführt wurde, s. (Adelson-Velsky & Landis 1962):

- AVL-Grid für tabellarischen Aufbau.
- AVL-Tree für hierarchischen Aufbau.

Es wurde die erste aufgeführte Darstellungsalternative implementiert, welche die Klasse *CL_GUI_ALV_GRID* verwendet. Diese hat grundsätzlich zwei wichtige Komponenten, die beide beim Erstellen einer Instanz als Parameter angegeben werden müssen: Zum einen dient die interne Tabelle *IT_OUTTAB* als Ausgabeliste zur Darstellung von relevanten Daten für den Anwender, zum anderen gibt der Parameter *IT_FIELDCATALOGUE* an, in welchem Format die Daten im ALVGrid angezeigt werden sollen. Zudem kann auch vorgegeben werden, wie auf einfache oder doppelte Klicks reagiert werden soll, indem die Ereignisse *DOUBLE_KLICK* bzw. *HOTSPOT_CLICK* im Eventhandler registriert werden (Keller & Krüger, 2006), welcher im nächsten Abschnitt ausführlicher erklärt wird.

Der Eventhandler
Soll ein Ereignis, welches vom ALV-Grid ausgelöst wird, abgefangen werden, so wird dieses durch den Eventhandler realisiert. Dieser basiert auf einem Publish-and-Subscribe-Mechanismus, was bedeutet, dass nur Eventhandler, die das Ereignis auch explizit registriert haben, auch darauf reagieren können (Keller & Krüger, 2006). In der Regel wird dieser als lokale Klasse definiert, welche dann statische Methoden besitzt (siehe Codebeispiel unten). So reagiert folgender Eventhandler beispielsweise nur auf Doppelklicks des ALV-Grids, wohingegen nur auf Drag Drop-Ereignisse des ALV-Trees eingegangen wird.

Implementierung des Programms

Aufruf des Programms

Alle Programme in SAP werden mittels eindeutigen Transaktionscodes aufgerufen, welcher in diesem Fall der Transaktionscode *Z3SD71* ist. Das Programm, an das die Transaktion anknüpft, heißt ebenfalls *Z3SD71*. Hierbei greift bereits die erste Berechtigungsprüfung ein, die den Programmzugriff durch unberechtigte Anwender verweigert:

```
1   zcl_check_auth=>tcode('Z3SD71').
2   CALL FUNCTION 'ZSDCC_VIEW_AST_KO'.
```

Der Befehl *zcl_check_auth=>tcode* überprüft, ob der aktuelle Anwender die Berechtigung für das Aufrufen dieser bestimmten Transaktion hat. Es erscheint eine Fehlermeldung, falls dies nicht zutrifft. Ist der Anwender zum Aufruf des Programms berechtigt, wird der Funktionsbaustein *ZSDCC_VIEW_AST_KO* aufgerufen, welcher wie eine Ausführung von separatem Programmcode gesehen werden kann, der Import- sowie Exportparameter besitzt und in beliebig vielen anderen Programmen wiederverwendet werden kann. Dieser initialisiert zunächst das globale Objekt *go_sdcc_text*, das zum Laden von Texten dient, die dann im späteren Programmablauf verwendet werden können. Schließlich wird die Benutzeroberfläche aufgerufen.

Gestaltung der grafischen Oberfläche

Titelleiste und Status
Je nach Sprache kann ein eigener Titel definiert werden. Dies wird im PBO-Modul mittels der Abfrage der Variable *sy-langu* ermittelt. Ist die Systemsprache deutsch, so wird der deutsche Titel angezeigt, andernfalls wird der englische Titel verwendet.

Des Weiteren wird der Status des Dynpros festgelegt. Hier werden folgende Elemente der Benutzeroberfläche definiert:

- Drucktastenleiste: Hier wird die Drucktaste mit der Funktion „Formular ausfüllen“ hinzugefügt (siehe Abbildung 6: Konfiguration der Drucktastenleiste):

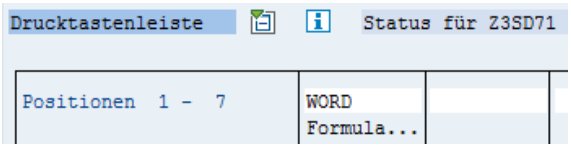

Abbildung 6: Konfiguration der Drucktastenleiste

Die erste Zeile legt fest, welcher Funktionscode beim Betätigen einer Drucktaste ausgelöst wird. Dieser kann nach Ausführen einer Aktion im PAI abgefragt und entsprechend behandelt werden. Die zweite Zeile zeigt an, wie die Drucktaste aussehen soll, d.h. ob diese Taste ein Symbol oder Text anzeigt. Für die Option eines Symbols gibt es eine Auswahl an bereits existierenden Bildern, während der Text frei wählbar ist.

- Funktionstastenleiste: Hier wird festgelegt, welche Funktion die von SAP bereits vorhandenen Funktionstasten haben sollen. Dabei muss nicht jeder Taste ein Funktionscode zugeordnet werden. Ist kein Code vorhanden, so wird dieses Symbol auf der Benutzeroberfläche ausgegraut. Der Funktionscode, der beim Betätigen einer Taste dann ausgelöst wird, kann dann im PAI Modul abgefangen und entsprechend behandelt werden (siehe Abbildung 7: Konfiguration er Funktionstastenleiste).

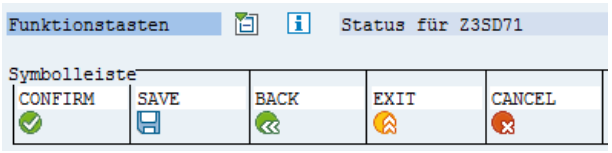

Abbildung 7: Konfiguration er Funktionstastenleiste

Für beide Alternativen wurden Tastenkürzel zugewiesen , welche den Schnellzugriff ermöglichen. Um das Eingabefeld für den Betriebsbeleg und den beschreibenden Text hinzuzufügen, wird der bereits im SAP integegrierte Screen Painter verwendet. Das Gesamtergebnis des Dynpros ergibt sich somit aus der Titel-, Drucktasten- und Funktionstastenleiste sowie den Elementen aus dem Screen Painter (siehe Abbildung 8: Ergebnis der Dynprokonfiguration):

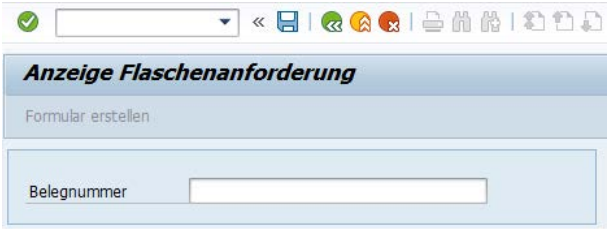

Abbildung 8: Ergebnis der Dynprokonfiguration

Die Generierung aller genannten Elemente erfolgt in einem PBO Modul.

Container

Um die Benutzeroberfläche sinnvoll aufzuteilen und um ALV-Grids und den Editor zuzuordnen, werden nach (Team, 2006) drei Container benötigt. Der erste ist ein Docking Container, der die Grundlage für den Splitter Container bildet :

```
IF go_body_docking_container
  IS NOT BOUND.
CREATE OBJECT go_body_docking-
                _container
  EXPORTING
        repid = sy-repid
        dynnr = sy-dynnr
        side = cl_gui_docking_container =>d
        ock_at_bottom
        ratio = 90
```

Wie in obiger Abbildung zu sehen ist, wird im Parameter „*side*“ das Attribut *dock_at_bottom* angegeben. Dies führt dazu, dass der Container sich an den unteren Rand anheftet, sodass das Eingabefeld für die Vertriebsbelegnummer nicht verdeckt wird. Ebenfalls wird beim Erzeugen von Objekten immer vorher geprüft, ob diese bereits instanziiert wurden, um eine wiederholte Erzeugung vorzubeugen. Ist dies nicht der Fall, so werden die Objekte mittels CREATE-Befehl erzeugt. Dieses Objekt dient nun als „Parent“ Container für den Splitter Container, der nach fast identischem CREATE-Befehl instanziiert wird. Danach werden weitere Teilcontainer erzeugt. Die beiden Parameter ROWS und COLUMNS bestimmen das Teilungsverhältnis des Splitter Containers.
rows = 1 und *columns = 3* ergibt somit drei Container, welche den „Parent“ Docking Container vertikal in der Mitte teilen. Der erste bildet die Grundlage für die Liste mit den Positionen, der zweite wird für die Liste mit zugehörigen Ausstattungen benötigt, der dritte wird vom Editor besetzt. Durch den Methodenaufruf *get_container* wird den Containern der Inhalt zugewiesen. Anschließend wird die Breite der Container mit der Methode *set_column_width* bestimmt, damit das Verhältnis der drei Container sinnvoll festgelegt ist.

Eventhandler

Auf ausgelöste Events muss entsprechend reagiert werden. Laut (Riekert, 2001) wird hierfür standardmäßig eine lokale Klasse, hier *lcl_event_receiver* genannt, definiert. Diese registriert Methoden für die jeweiligen Ereignisse, welche abgefangen werden sollen:

```
CLASS lcl_event_handler DEFINITION .
  PUBLIC SECTION.
  METHODS:
    handle_hotspot_click FOR EVENT
      hotspot_click OF cl_gui_alv_grid
     IMPORTING e_row_id ,

    handle_data_changed FOR EVENT
      data_changed OF cl_gui_alv_grid
      IMPORTING er_data_changed
                e_ucomm
                s ende r ,
```

```
11   handle_function_selected
12     FOR EVENT function_selected OF
         cl_gui_toolbar
13     IMPORTING fcode.
14 ENDCLASS. "lcl_event_handler DEFINITION
```

Wie im obigen Code zu erkennen ist, werden drei Ereignisse durch Methoden abgefangen:

- *hotspot_click:* Dieses wird jedes Mal vom linken ALV-Grid ausgelöst, wenn auf eine Position geklickt wird. Dadurch wird die interne Tabelle für das rechte ALV-Grid geleert und mit den Datensätzen der angeklickten Position gefüllt. Es folgt dann die Aktualisierung der Tabellenanzeige durch den Befehl refresh_table-_display.
- *data_changed*: Dieses wird jedes Mal vom rechten ALV-Grid ausgelöst, wenn manuell ein Eintrag getätigt wurde. Dadurch wird dann vorgemerkt, dass eine Änderung vorgenommen wurde. Wenn der Benutzer dann die Trans-aktion verlassen oder ins vorherige Dynpro zurückspringen will, so wird vorher gefragt, ob die Änderungen gespeichert werden sollen oder nicht.
- *function_selected*: Dieses wird jedes Mal vom rechten ALV-Grid ausgelöst, wenn die Drucktaste „Eintrag hinzufügen" betätigt wird. Im Editor werden dann zwei neue Zeilen eingefügt. Die erste Zeile enthält den Namen des Änderers, das Datum und die Uhrzeit des Änderungszeitpunkts. Die zweite Zeile ist eine Leerzeile, die dem Benutzer die Möglichkeit gibt, Informationen zu den Vertriebsbelegen hinzuzufügen, die relevant für die Kundenobjekte sind.

Des Weiteren wird angegeben, welche Ereignisse von welchen Objekten abgefangen werden sollen:

```
1 SET HANDLER go_event_handler->handle–
    _function_selected FOR go_toolbar.
2 SET HANDLER go_event_handler->handle-
    _hotspot_click FOR go_alv_grid_pos.
3 SET HANDLER go_event_handler->handle-
    _data_changed FOR go_alv_grid_obj–
                        _ko.
```

Feldkatalog

Da nun die Container erzeugt wurden, können diesen ALV-Grids zugewiesen werden. Jedoch muss spätestens beim Anzeigebefehl der Liste ein Feldkatalog mitgegeben werden, der mit der Struktur der internen Tabelle übereinstimmt, damit eine Anzeige erfolgen kann. Die Generierung erfolgt durch den bereits vorhandenen Funktionsbaustein *LVC_FIELDCATALOG-_MERGE* von SAP. Durch diesen wird festgelegt, welche Spalten in welchem Format angezeigt werden sollen. Übergeben wird die Struktur *ZSDCC_MANA-GER_LIST* (siehe Tabelle 2: Aufbau der Struktur ZSDCC_MANAGER_LIST) als Export Parameter. Dabei wurde diese bereits im Voraus als Dictionary-Objekt angelegt.

Feld	Datentyp	Kurzbeschreibung
eq_manager	Objekt	Equipment Manager
ast_si_manager	Objekt	Ausstattung Manager
ko_manager	Objekt	Kundenobjekt
posnr	NUMC	Manager
arktx	CHAR	Positionsnummer
		Beschreibung

Tabelle 2: Aufbau der Struktur ZSDCC_MANAGER_LIST

Der Feldkatalog für die zweite Liste wird anhand der Struktur *ZSDCC_AST_KO* (siehe Tabelle 4.2: Aufbau der Struktur *ZSDCC_AST_KO*) auf dieselbe Weise erzeugt.

Feld	Datentyp	Kurzbeschreibung
ast_nr	CHAR	Ausstattung Nummer
ko	String	Kundenobjekt
objnr	NUMC	Kundenobjekt Nummer
matnr	CHAR	Materialnummer
ko_bez	String	Kundenobjekt Bezeich-
ko_num	String	nung
posnr	NUMC	Preform
		Positionsnummer

Tabelle 3: Aufbau der Struktur ZSDCC_AST_KO

Dadurch wird ein Feldkatalog zurückgegeben, welcher alle Spalten mit ihren jeweiligen Eigenschaften in Form einer internen Tabelle enthält. Sobald der Feldkatalog generiert wurde, kann eine Schleife auf die interne Tabelle angewendet werden, um Spalteneigenschaften zu modifizieren. Für den Feldkatalog mit den Positionen werden nur die Spalten mit der Positionsnummer und deren Beschreibung angezeigt, während die restlichen Spalten ausgeblendet werden. Zudem reagiert jede Positionsnummer auf Klicks und löst ein Ereignis aus, das später durch einen Eventhandler behandelt werden kann.Auch die Spaltenüberschriften können durch die Felder *scr_text_l*, *scr_text_m* und *scr_text_s* bestimmt werden. Dabei wird zwischen drei verschiedenen Textlängen unterschieden (long, medium, short), um je nach aktueller Spaltenbreite einen passenden Text anzuzeigen.

ALV-Grid

Die Instanziierung von ALV-Grids erfolgt durch einen CREATE-Befehl, bei dem der zugehörige Container angegeben wird. Ergänzend zum Feldkatalog, der Spalteneigenschaften enthält, kann auch ein Layout in Form einer weiteren Struktur mitgegeben werden, wodurch mittels Anpassung von bestimmten Feldinhalten folgende Modifikationen vorgenommen werden
können:

- *grid_title*: Legt den Titel über dem ALV-Grid fest.
- *no_toolbar*: Blendet die Toolbar – die standardmäßig integriert ist – aus.
- *col_opt*: Optimiert die Spaltenbreite basierend auf den Inhalt.

Wurden alle Einstellungen vorgenommen, so kann anschließend die Methode *set_table_for_first_display* zur Anzeige der Liste aufgerufen werden. Dabei wird noch eine interne Tabelle dem Parameter *it_outtab* übergeben,

um vorzugeben, welchen Inhalt die Listausgabe anzeigen soll.

Freitext Editor
Damit der Anwender die Möglichkeit hat, Freitext zu verfassen, muss dem letzten Container noch ein Editor zugewiesen werden. Dieser ergibt sich aus der Klasse *CL_GUI_TEXTEDIT* und wird ebenfalls durch einen CREATE-Befehl instanziiert. Da eine Eingabe zum Zeitpunkt des Programmstarts keinen Sinn ergeben würde, wird der Editor mittels Methode *set_read-only_mode* in den Lesemodus gesetzt. Diese Sperre kann im späteren Verlauf wieder aufgehoben werden. Der Text wird jedes Mal, wenn eine neue Belegnummer
gelesen wird, aus der Tabelle *ZTM_KONTORDER_K* mit dem Vertriebsbeleg als Primärschlüssel gelesen. Da dieser allerdings im XML-Format gespeichert wird, muss vorerst mittels Befehl CALL TRANSFORMTION konvertiert werden:

```
1 CALL TRANSFORMATION i d
2         SOURCE XML gwa_contorde r_k-d e s c r
3         RESULT r o o t = wa _ t e x t t a b l e .
```

Das Ergebnis ist eine interne Tabelle, die dem Freitext Editor als Importparameter mittels Methode *set_text_as-_stream()* übergeben werden kann. Beim Abspeichern des Textes verläuft der ganze Prozess rückwärts, was heißt, dass der Text zuerst in eine interne Tabelle gelesen, in XML-Format konvertiert und dann in die Datenbank geschrieben wird. Falls bereits ein Eintrag zum Vertriebsbeleg existiert, so wird dieser gelöscht und ein neuer Eintrag hinzugefügt.
Nach Initialisierung aller vorher erläuterten Objekte ist das Gesamtergebnis des Dynpros in Abbildung 9 zu sehen.

Abbildung 9: Gesamtergebnis des Dynpros

Z3SET Parameter

Änderungen am Verhalten eines Programms, wie zum Beispiel welche Daten gelesen werden, können aufwendig sein, da jedes Mal neu in das Produktivsystem transportiert werden muss. Um diesen Aufwand zu vermeiden, existiert die Transaktion *Z3SET*. Diese kann verwendet werden, um Parameter anzulegen, die von allen Programmen ausgelesen werden können. Die Parameter lassen sich auch in jedem System unabhängig voneinander nach Belieben anpassen, sodass Transporte für Änderungen dieser Art überflüssig werden. Daraus resultieren dynamischere Programme, deren Ablauflogik aufgrund der Verwendung flexibler Parameter leichter anpassbar sind als die von herkömmlichen Programmen. Dabei gibt es drei verschiedene Parameter-Typen: boolean, numerisch oder Zeichenkette.
Für dieses Projekt wird ein Parameter angelegt (siehe Abbildung 10:Z3SET Parameter SD.Z0SD16.MATNR. Dieser legt fest, welche Vertriebsbelegpositionen bei der Selektion aufgrund ihrer Materialnummer ausgelesen werden sollen. Der Parametername lautet *SD.Z0SD16.MATNR* und ist eine Stringvariable, die relevante Materialnummern für die Flaschenanforderung speichert, die durch Strichpunkte getrennt sind. Damit wird im Programm vorgegeben, welche Merkmale genau aus einer internen Tabelle ausgelesen werden sollen, da nur spezielle Eigenschaften für diesen Zweck benötigt werden.

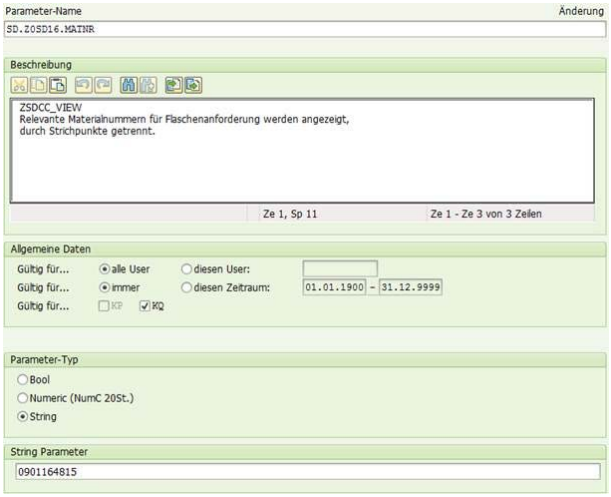

Abbildung 10:Z3SET Parameter SD.Z0SD16.MATNR

Datenselektion

Wurde die Vertriebsbelegnummer eingegeben und bestätigt, so springt das Programm in das PAI Modul. In diesem läuft der Verarbeitungsprozess ab, der nicht durch den Eventhandler behandelt wurde. Dazu gehört die Identifikation des Funktionscodes, welcher durch eine Aktion ausgelöst wurde. Dieser wird anhand der Systemvariable *sy-ucomm* ermittelt. Enthält dieser den Wert *CONFIRM*, so wird die Datenselektion gestartet.
Diese unterteilt sich – wie in der Unterteilung des Dynpros – in drei wichtige Teilschritte:

- Positionen und Equipments.
- Ausstattungen und Kundenobjekte.
- Flaschenanzahl.

Dabei werden Schleifen auf Listen der Equipments, Ausstattungen und Kundenobjekte verwendet, um alle Bestandteile von allen jeweiligen Positionen zu erhalten. Wurden alle benötigten Daten, ausgehend von der Position bis hin zu den zugehörigen Kundenobjekten, gesammelt, so werden, basierend auf diesen Informationen, die benötigte Flaschenanzahl ermittelt und der zusammengefasste Datensatz zur Ergebnisliste hinzugefügt.
Abschließend wird der Freitext des angegebenen Vertriebsbeleg aus der Datenbanktabelle *ZTM_CONT-ORDER_K* ausgelesen. Das Struktogramm in Abbil-

dung 11 veranschaulicht den groben Ablauf – die Darstellung orientiert sich an einer entsprechenden in (Balzert 2011):

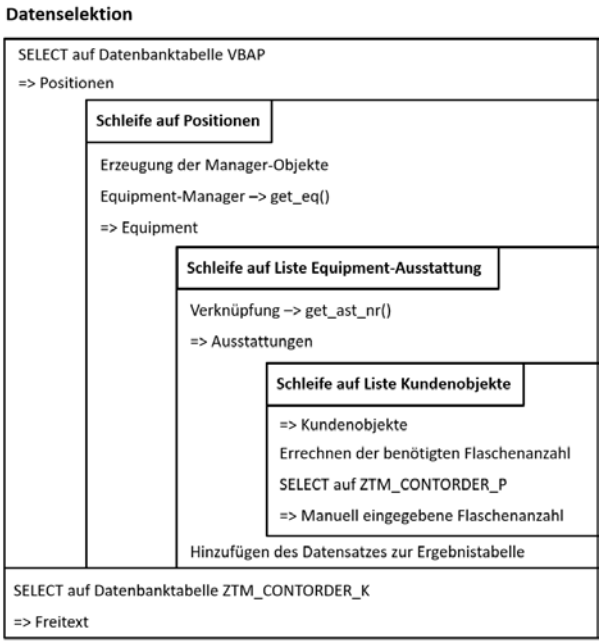

Abbildung 11: Struktogrammn zum Ablauf der Datenselektion

Die folgenden Abschnitte erläutern die einzelnen Schritte noch einmal genauer.

Positionen und Equipments
Als erstes müssen anhand des Vertriebsbelegs alle zugehörigen Positionen ermittelt werden. Dies wird durch das Lesen auf die Datenbanktabelle *VBAP* realisiert, wobei das Selektionskriterium die Vertriebsbelegnummer ist. Da die Positionen einzeln ausgelesen werden, wird dieser Vorgang in Form einer Schleife ausgeführt, um für jeden Datensatz auch gleich die restlichen Informationen auszulesen, bevor die nächste Position ermittelt wird. Mit der Positionsnummer der aktuellen Schleife werden die Manager-Objekte erzeugt, um nicht nur das zugehörige Equipment zu erhalten, sondern auch die Ausstattungen und Kundenobjekte, die für die nächsten Schritte erforderlich sind:

```
1 CALL METHOD go_sdcc_data->create_
                                  _objects
2   IMPORTING
3     eq_manager = wa_managers-eq_manager
4     ast_si_manager = wa_managers-ast_si-
                                      _manager
5     ko_manager = wa_managers-ko_manager .
```

Ausstattungen und Kundenobjekte
Ausgehend vom Equipment-Manager kann eine Schleife auf die interne Tabelle des Equipment-Objekts ausgeführt werden. Damit werden die zugehörigen Ausstattungsobjekte mittels der Objekte, die eine Verknüpfung zwischen Equipment und Ausstattungen darstellen, ermittelt. Ist die Ausstattungsnummer in einer Struktur – in ABAP auch Workarea genannt – vorgemerkt, so wird dann auf die Liste mit Kundenobjekten zugegriffen. Somit werden die Objektnummer, Bezeichnung und Materialnummer des Behälters sowie der Preform gelesen und ebenfalls zur Workarea hinzugefügt. Die Namen der Felder, die gelesen werden, ergeben sich aus dem Z3SET Parameter *SD.Z0S-D16.MATNR*. Dieser enthält einen String, in denen die relevanten Feldbezeichnungen der Eigenschaften durch Strichpunkte getrennt sind. Durch einen Split-Befehl werden die einzelnen Komponenten in eine interne Tabelle *lt_chars* geschrieben. Danach wird auf die *lt_chars* eine Schleife ausgeführt, in der die einzelnen Zeichenketten der Range-Tabelle *crt_matnr* hinzugefügt werden (siehe Listing 10: Befüllen der Range-Tabelle). Diese dient als Filter bezüglich des Auslesens von Informationen der Kundenobjekte. Hierbei ist zu beachten, dass dieses Objekt übersprungen wird, wenn es sich um keine PET-Flasche als Behältertyp handelt:

```
1 LOOP AT lt_chars ASSIGNING <wa_char>.
2   wa_matnr-low = <wa_char>.
3   wa_matnr-sign = 'I'.
4   wa_matnr-option = 'EQ'.
5   APPEND wa_matnr TO crt_matnr.
6 ENDLOOP.
```

Flaschenanzahl
Um die Summe von benötigten Testmaterialien zu erhalten, wird zunächst auf die Tabelle *ZTM_AUSTATT* mit dem Vertriebsbeleg zugegriffen. Dabei muss die Identifikationsnummer '1' sein, welches Kennzeichen für „Behälter" als Kundenobjekttyp ist, um nur Datensätze auszulesen, bei denen es sich um Behälter handelt. Sind Datensätze vorhanden, so wird mit der gleichen Vertriebsbelegnummer und Objektnummer auf die Datenbanktabelle *ZFLASCHEN* zugegriffen, um die Materialnummer zu erhalten. Falls diese in der Liste des Kundenobjekt-Managers vorhanden ist, dann wird die Flaschenanzahl mittels der Informationen der vorherigen Tabelle *ZTM_AUSTATT* ausgerechnet.
Das Ergebnis ist das Produkt von der Paketanzahl und der einzelnen Stückzahl der Position und wird zur Gesamtsumme, die für jede Objektnummer separat vorhanden ist, addiert. Sobald die Berechnung abgeschlossen ist, wird das Ergebnis der Workarea hinzugefügt, die daraufhin zur Ergebnistabelle angehängt wird.
Das Endergebnis nach der Datenselektion wird dann im Dynpro dargestellt (siehe Abbildung 12: Ergebnislisten nach Datenselektion).

Abbildung 12: Ergebnislisten nach Datenselektion

Manuelle Eingabe der Flaschenanzahl

In manchen Fällen kann es sein, dass die errechnete Flaschenanzahl für spezielle Situationen nicht geeignet ist. Deshalb wird neben der Spalte der berechneten Zahl eine

weitere Spalte angeboten (siehe Abbildung 12: Ergebnislisten nach Datenselektion), die es dem Benutzer ermöglicht, manuelle Eingaben zu tätigen. Falls eine manuelle Änderung vorgenommen wurde, dann wird diese auch beim Befüllen des Formulars berücksichtigt. Statt der errechneten Anzahl wird somit die benutzerdefinierte Zahl bei Erstellen des Word-Dokumentes verwendet.
Wenn die Transaktion geschlossen wird oder in das vorherige Dynpro zurückgesprungen werden soll, erscheint vorher ein Abfrage, ob die Benutzereingaben gesichert werden soll. Grund hierfür ist das Event *data_changed*, das durch den manuellen Eintrag ausgelöst wird. Dieses Ereignis wird durch den Eventhandler abgefangen, welcher daraufhin die globale Variable *gv_data_changed* auf einen wahren Wert setzt. Da bei jedem Dynpro- oder Vertriebsbelegwechsel geprüft wird, ob Daten abgeändert wurden, erscheint ein Popup-Fenster, das den User fragt, ob die Änderungen gespeichert oder verworfen werden sollen. Sobald der Benutzer diese Anfrage bestätigt, werden die manuell eingegeben Zahlen in die Datenbank *ZTM_CONTOR-DER_P* geschrieben, sodass diese bei der nächsten Anzeige von genau diesen Kundenobjekten auch aufgelistet werden. Die Datenbanktabelle sichert zudem auch alle weiteren Informationen, die für die eindeutige Identifizierung des veränderten Eintrags benötigt werden.

Erstellen, Befüllen und Senden des Formulars

Nachdem alle Daten überprüft und unter Umständen ausgebessert wurden, kann die Flaschenanforderung an die nächste Abteilung weitergeleitet werden. Diese Funktion übernimmt die Funktionstaste „*Formular erstellen*“ (siehe Abbildung 9: Gesamtergebnis des Dynpros), die bei Betätigung alle Daten in einer internen Tabelle sammelt, die in das Word-Dokument geschrieben werden sollen. Dazu werden zunächst Empfängername und Absender, sowie die Email-Adressen beider Personen in eine Workarea geschrieben und der Tabelle hinzugefügt. Im Anschluss werden die Spaltenüberschriften definiert, bei dem die Texte aus dem Objekt *go_sdcc_text* gelesen werden. Danach erfolgt eine Schleife über die interne Tabelle *gt_obj_ko*, um alle Kundenobjekte und zugehörige Informationen anzuhängen:

```
1 LOOP AT gt_obj_ko ASSIGNING
    <fs_obj_ko>.
2   CLEAR: wa_cell, wa_line.
3   wa_cell-text-value = <fs_obj_ko>-
                              objnr.
4   SHIFT wa_cell-text-value LEFT DELETING
      LEADING '0'.
5   APPEND wa_cell TO wa_line-cells.
6   ...
7   IF <fs_obj_ko>-manuell IS INITIAL.
8     wa_cell-text-value = <fs_obj_ko>-
                               count.
9   ELSE.
10    wa_cell-text-value =
        <fs_obj_ko>-manuell.
11  ENDIF.
12  SHIFT wa_cell-text-value LEFT DELETING
      LEADING '0'.
13  APPEND wa_cell TO wa_line-cells.
14  APPEND wa_line TO wa_table-lines.
15 ENDLOOP.
```

Dabei werden nicht nur die Kundenobjekte der aktuellen Position in das Formular geschrieben, sondern alle Kundenobjekte des Vertriebsbelegs.
Nach Speicherung aller Daten in der internen Tabelle wird diese im xml-Format gespeichert und dem Objekt *lo_word* als Parameter übergeben. *lo_word* ist eine Instanz der Klasse *zcl_ooxml_word* und wird für das Senden des Word-Formulars verwendet. Daraufhin wer-den unter Anderem noch folgende Informationen als Control-Daten in der Workarea *wa_controldata* übergeben:

- *templateid:* Festlegung, welche Word-Vorlage verwendet wird.
- *operatormail:* Email-Adresse des Senders.
- *operatorname:* Name des Senders.
- *language:* Sprache der Email.
- *knummer:* Benutzerkennung.

Diese werden dem Objekt ebenfalls als Parameter übergeben. Es folgt dann der Methodenaufruf send(), woraufhin das Formular (vgl. Abbildung 13: Ausgefülltes Word-Dokument) befüllt und an den Benutzer gesendet wird. Dabei ist die Datei nicht dem Anhang der Email beigefügt, sondern muss durch Klick auf einen in der Email enthaltenen Link heruntergeladen werden. Grund dafür ist, dass das fertige Formular sich an einem bestimmten Ablageort befindet, welcher für einige Tage abgerufen werden kann.

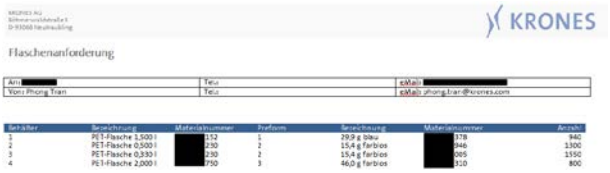

Abbildung 13: Ausgefülltes Word-Dokument

Testphase

Für das Testen der neuen Transaktion werden Testdaten benötigt. Da diese im Entwicklungssystem unzureichend vorhanden sind, wurde das Programm in das Qualitätssicherungssystem transportiert. Dies erfolgt durch den Transport Organizer, der alle Bestandteile der neuen Software in Form eines Sammelauftrags zusammenfasst und diese dann vom Entwicklungssystem KE1 ins Qualitätssicherungssystem KQ1 transferiert. Die Tests wurden sowohl vom Entwickler, als auch von den involvierten Fachbereichen durchgeführt.

Berechtigungsprüfung
Als erstes wurde getestet, ob der Aufruf der Transaktion nur bei vorhandener Berechtigung ausgeführt wird. Dieser Vorgang verlief einwandfrei. Benutzer, denen die Berechtigung zugewiesen wurde, können das Programm aufrufen, während unberechtigte Personen eine Fehlermeldung bekommen.

Korrekte Anzeige der Dynpro-Elemente
Sowohl die Funktions- als auch Drucktastenleiste müssen alle definierten Komponenten darstellen und bei Betätigung die dazu gehörige Funktionalität ausführen können. Die ALV-Grids und der Freitext-Editor müssen mit ihren Daten angezeigt werden. Auch die Toolbars der ALV-Grids sollen ausgeblendet sein. Das Docking-Verhalten der verschiedenen Container wurde ebenfalls überprüft, um sicherzustellen, dass die Dynpro-Elemente sich nicht gegenseitig überlappen.

Funktionalität der Dynpro-Elemente
Alle Buttons müssen ihre zugewiesene Funktionalität bei Klick ausführen können. Dazu gehören folgende Komponenten:
- Tasten der Funktionstastenleiste.
- Drucktaste "*Formular erstellen*".
- Drucktaste "*Eintrag hinzufügen*".

Des Weiteren müssen die zugewiesenen Tastenkürzel auch die geforderte Funktionalität aufweisen.

Korrektheit der Daten
DesWeiteren wurde überprüft, ob bei Eingabe eines Vertriebsbelegs folgende Daten richtig ausgelesen werden:
- Positionsdaten des Vertriebsbelegs.
- Kundenobjekte der Positionen.
- Behälter- und Preformdaten.
- Flaschenanzahl.
- Freitext.

Zusätzlich wurde getestet, ob diese Informationen auch korrekt im Word-Dokument abgebildet werden.

Reaktion auf Single-Clicks im ALV-Grid
Es folgten Klicks auf verschiedene Positionsnummern, um zu überprüfen, ob eine Aktualisierung der ALV-Grids erfolgt. Wichtig hierbei war, dass die Kundenobjektliste bei jeder Aktualisierung nur die zugehörigen Kundenobjekte der markierten Position anzeigen. Falls bei der Flaschenanzahl manuelle Einträge zu Kundenobjekten vorhanden sind, müssen diese auch automatisch in das passende Feld geschrieben werden.

Möglichkeit für Benutzereingaben
Sowohl die Spalte für die manuelle Eingabe der Flaschenanzahl als auch das Eingabefeld des Freitext-Editors sollen Benutzereingaben entgegennehmen können. Alle restlichen Spalten der ALV-Grids sollen nicht editierbar sein. Die Möglichkeit, Eingaben im Freitext-Editor ohne vorhandenem Vertriebsbeleg zu tätigen, soll verweigert werden.

Registrierung von Änderungen bzw. Benutzereingaben
Falls Benutzereingaben getätigt wurden, soll das Programm diese Aktion vormerken, sodass im späteren Verlauf ein Hinweis auf Speicherung der Änderungen aufgerufen werden kann. Vier Fälle wurden durch die Tests abgedeckt:
- Manuelles Hinzufügen von benutzerdefinierter Flaschenanzahl.
- Manuelles Abändern von bereits vorhandener benutzerdefinierter Flaschenanzahl.
- Hinzufügen von Text im Freitext-Editor.
- Abändern von Text im Freitext-Editor.

Anzumerken ist auch, dass die Registrierung wieder zurückgenommen wird, wenn der Speicherbutton betätigt wird.

Speicherung von Änderungen bzw. Benutzereingaben
Benutzer sollen die Möglichkeit haben, ihre Änderungen abzuspeichern, sodass diese in späteren Aufrufen weiterhin bestehen. Somit wurde getestet ob bei Betätigung des Speicherbuttons alle Änderungen in die zugehörigen Datenbanken geschrieben werden.
Des Weiteren wurde überprüft, ob das Popup zur Speicherung der Daten erscheint, wenn User Änderungen vorgenommen haben und ohne Speicherbefehl
- die Transaktion beenden,
- eine andere Belegnummer eingeben oder
- in das vorherige Dynpro zurückspringen

wollen.
Wird auf OK geklickt, so soll, wie im Fall des Speicherbuttons, der Speichervorgang initiiert werden.

Emailfunktion
Bei diesen Tests wurde überprüft, ob das fertige Formular an den Benutzer gesendet wird. Zusätzlich müssen die Informationen von Sender und Empfänger stimmen. Wichtig ist bei dieser Funktion auch, dass nicht nur die Informationen von Kundenobjekten der markierten Position in das Dokument übernommen werden. Auch alle Datensätze von relevanten Positionen des gleichen Vertriebsbelegs sollen transferiert werden.

Durch die Durchführung der oben genannten Funktionstest wurde sichergestellt, dass alle funktionalen Anforderungen erfüllt sind.

Erfüllung nichtfunktionaler Anforderungen
Im Verlauf der Testphase wurde das Programm auch auf die nichtfunktionalen Anforderungen geprüft:
Die Anwenderfreundlichkeit ergibt sich durch den übersichtlichen und intuitiven Aufbau der Benutzeroberfläche. Zudem wird der Benutzer mithilfe von Nachrichten – in Form von Dialogfenster oder Meldungen in der Statusleiste – über alle Handlungen des Programms informiert.
- Durch objektorientierte Programmierung sind Änderungen bzw. Erweiterungen einfach durchführbar. Auch die Verwendung vom Z3SET-Parameter fördert die Wartbarkeit.
- Eine gute Performance wird dadurch gewährleistet, dass nur wenige Datenbankzugriffe vom Programm durchgeführt werden. Erkennbar ist dies durch kurze Wartezeiten nach Eingabe des Vertriebsbelegs. Auch alle anderen Funktionalitäten werden ohne größere Verzögerungen ausgeführt.

- Das Programm reagiert zuverlässig auf alle möglichen Benutzereingaben und gibt auch keine unerwarteten Fehlermeldungen aus. Auch alle Funktionalitäten werden ordnungsgemäß ausgeführt.

Mit dem Ergebnis, das sowohl die funktionalen, als auch die nichtfunktionalen Anforderungen erfüllt werden konnten, wurde die Testphase beendet. Dies wurde in Form eines Abnahmeprotokolls, das von den Fachbereichen unterschrieben wurde, festgehalten. Mit dem Beenden dieses Vorgangs ist, aus Sicht der Fachbereiche, das Programm bereit für den Transport in das Produktivsystem.

Zusammenfassung

Durch dieses Projekt wurde eine Lücke bei der umfassenden Digitalisierung der Geschäftsprozesse im Hause Krones geschlossen. Die Testspezifikation erfolgt nun basierend auf aktuellen Produktions- und Kundendaten im SAP System und ist Teil des SAP Systems bei Krones. Damit werden vorallem zwei Verbesserungen erzielt:

- qualitativ höherwertige Testspezifikationen und
- drastisch reduzierter manueller Anteil am Arbeitsaufwand mit einer jährliche Zeitersparnis von ca. 430 Stunden.

Bei der hier betrachteten Komplettanlage mit Blasmaschine handelt es sich um ein wichtiges Produkt von Krones, aber nicht um das einzige. Für andere Flaschenanlagen sind ähnliche Testspezifikationen zu erstellen, so dass dieses Vorgehen in der nahen Zukunft übertragen werden wird. Ferner trägt der Erfolg des Projekts dazu bei, verstärkt Digitalisierungsmöglichkeiten zu suchen und durchzuführen.

Literatur

Adelson-Velsky, G. M. & Landis, E. M. (1962): *Один алгоритм организации информации*. In: *Doklady Akademii Nauk SSSR*. 146, 1962, S. 263–266. (russisch)
Englische Übersetzung von Myron J. Ricci: An algorithm for the organization of information. In: Soviet Mathematics 3, 1962, S. 1259–1263.

Balzert, H. (2011): Lehrbuch der Objektmodellierung: Analyse und Entwurf mit der UML 2. Heidelberg: Spektrum Akademischer Verlag.

Gadatsch (2015). Geschäftsprozesse analysieren und optimieren (essentials). Bonn: Springer Vieweg.

Keller, H. & Krüger, S. (2006). *ABAP Objects: ABAP-Programmierung mit SAP NetWeaver.* Bonn: Galileo Press.

Krones AG. (6. September 2016). *Geschäftsbericht Krones AG 2015.* Von Krones: http://www.krones.com/de/investor_relations/geschaeftsbericht-krones-ag-2015.php abgerufen.

Oestereich, Bernd (2009): Analyse und Design mit UML 2.3: Objektorientierte Softwareentwicklung. München: Oldenbourg Wissenschaftsverlag.

Riekert, R. (2001). *ABAP – Programmierung, Fortgeschrittene Programmiertechniken für ABAP.* München: Addison-Wesley Verlag.

Team, D. S. (2006). *SAP R/3 Black Book.* Neu Delhi: Dreamtech Press.