Digitale Transformation in Echtzeit: Die Ziele von morgen basierend auf dem Datenmodell von gestern

Merveille Nangmo Wanko BSc.[1]
Bernhard Zeller[1]
Professor Dr. Frank Herrmann[2]
[1]Maschinenfabrik Reinhausen
[2]Ostbayerische Technische Hochschule Regensburg
E-Mail: [1]nwmerveille@yahoo.fr, [1]b.zeller@reinhausen.com und [2]Frank.Herrmann@OTH-Regensburg.de

SCHLÜSSELWÖRTER
Digitale Transformation, Strategie, ALV-GRID, Dynpro

ABSTRACT

Die Digitale Transformation fordert Unternehmen aller Couleur. Ironischer Weise sind es gerade die bisher verwendeten IT-Systeme mit ihren starren Strukturen, die Unternehmen in Ihrer digitalen Transformation oft ausbremsen. Auch wenn die Softwarehersteller längst reagiert haben und neue, flexiblere Versionen ihrer Produkte anbieten, so ist ein größerer Softwarewechsel immer noch eine Herausforderung für Unternehmen und ein Schritt der wohlüberlegt und geplant sein will.
In dieser Arbeit wird deshalb ein Vorgehen vorgestellt, um mittels In-Memory Technologie und Virtualisierung zumindest die wichtigsten Ergebnisse der Transformation bereits auf den bestehenden Datenmodellen in Echtzeit zu generieren. Dadurch wird genug Zeit gewonnen, um die eigentliche Transformation der IT-Landschaft geplant und mit der notwendigen Sorgfalt durchzuführen.

Digitale Transformation als Treiber

Die zunehmende Digitalisierung revolutioniert unseren Alltag – sowohl im privaten wie im geschäftlichen Umfeld. In immer kürzeren Zeitabständen tun sich Chancen auf und immer schneller tauchen neue Mitbewerber auf.
Um diese Digitale Transformation zu meistern reicht es nicht aus auf diese Veränderungen zu reagieren. Um Erfolg zu haben müssen die Unternehmen diesen Wandel aktiv mitgestalten.
Die Maschinenfabrik hat deshalb schon früh begonnen sich mit den neuen Technologien und den dafür notwendigen Prozessen und Organisationsformen zu beschäftigen und damit die Digitale Transformation einzuleiten.
Eine solche Transformation eines weltweit operierenden Unternehmens ist aber kräftezehrend und in der Natur der Sache bedingt immer zu langsam, zumal die Ziele zu Beginn oft noch unscharf sind.
Um hier aktiv zu gestalten und nicht Getriebener zu werden ist es notwendig das Heft des Handelns in der Hand zu behalten. Im Falle der Maschinenfabrik Reinhausen heißt das sehr wohl Diversifikation und Indeenvielfalt zu fördern, als wichtige Quellen für wertvollen Input. D.h. aber auch zum richtigen Zeitpunkt die Kräfte wieder zu sammeln und auf einige wesentliche Kernthemen zu fokussieren, um diese wichtigen Themen voranzutreiben.
Der technische, in der IT Umsetzung verwendete Begriff für diese Kernthemen ist bei der Maschinenfabrik Reinhausen „*Strategische Geschäftsfelder"*, weshalb in der späteren Beschreibung der techn. Umsetzung dieser Begriff verwendet wird. Strategische Geschäftsfelder ist aber eher ein technischer Arbeitstitel. Im Folgenden werden wir aber beim Begriff Kernthema bleiben, da er besser passt.
Kernthemen haben wenig mit der vorhanden Unternhemensstruktur zu tun haben. Es sind die Antworten der MR auf die sich ständig wandelnden Bedürfnisse, sowohl aus Marktsicht, wie auch aus technologischer Sicht. Dabei muss nicht immer die Umsatzgenerierung im Vordergrund stehen, sondern es kann auch der gezielte Erkenntnisgewinn in manchen Bereichen sein.
Auszeichnungen wie der Industrie 4.0 Award, der Red Dot Design Award und die Erfolge bei den Great Place to Work Umfragen zeigen, dass dieses Werkzeug auf allen Ebenen greift – Technologisch, wie auch Organisatorisch.
Zunehmend wird aber deutlich, dass die IT-technische Umsetzung dieses Konzeptes an Grenzen stößt. Um auch hier agieren zu können statt zu reagieren sind moderne Lösungen notwendig.

IT Schifffahrt: Schnellbote und Tanker

Die Digitalisierung beschleunigt alles und verlangt immer mehr Flexibilität und Umsetzungsgeschwindigkeit. Ironischer weise sind es gerade die IT-Systeme, die in den Unternehmen genau diese Digitalisierung ausbremsen.
Die in heutigen Unternehmen eingesetzten ERP Systeme wurden in den achtziger und neunziger Jahren entwickelt und waren dafür konzipiert ein stabiles Geschäftsmodell abzubilden und tausende, manchmal sogar millionen von Transkationen in kurzer Zeit abzuarbeiten. Es waren also eher Tanker, die vorgefertigte Routen befuhren und dabei Unmengen von Rohstoffen beförderten. Ein Richtungswechsel war aufwändig - vom Bau eines neuen Tankers ganz zu schweigen.
Heute sind eher Schnellbote gefragt, die schnell gebaut sind und schnell neu ausgerichtet. Sollte doch mal mehr Transportkapazität gefragt sein, so wird diese kurzfristig gemietet, z.B. als Cloud Service.
Die Softwarehersteller haben längst reagiert und bieten mittlerweile neue ERP Systeme an, die genau dieses

Bedürfnis nach Flexibilität befriedigen. Allerdings ist für ein Unternehmen der Umstieg von einer Tankerflotte auf eine Flotte von Schnellboten nicht mal eben schnell gemacht. Das Bedarf der Planung und Vorbereitung um den täglichen Betrieb nicht zu stören.
Um diesen gordischen Knoten zu durchschlagen hat die MR nach Wegen Gesucht die wichtigsten Aspekte der Kernthemen auf der bestehenden IT-Landschaft abzubilden und so Zeit für den Umbau der IT-Landschaft zu gewinnen.

Das virtuelle Produkt

Um zumindest die wichtigsten Aspekte der Kernthemen mit dem Datenmodell der bisher genutzten IT Systeme abbilden zu können ist eine Umformung der Daten notwendig. Diese Umformung muss beliebig änderbar sein, in Echtzeit geschehen und bis zu einem gewissen Grad robust gegen Änderungen des darunter liegenden Datenmodells sein.
Im Prinzip will man ähnlich wie beim Kochen verschiedenste Zutaten (Elemente des darunter liegenden Datenmodells) immer wieder zu neuen Gerichten (Aspekte der verschiedenen Kernthemen) kombinieren und vielleicht auch mal eine vorhandende Zutat ändern (geraspelt statt geschnitten), neue Zutat hinzufügen und auch mal die Zubereitungsart ändern. Im Vordergrund steht also das Rezept, das aus den Zutaten was Leckeres macht.
In der IT Welt ist dieser Gedanke des Rezeptes in Form der Konfiguration von System und Produkten weit verbreitet. Insbesondere in der Konfiguration von Produkten besitzt die MR tiefes Wissen. Es lag deshalb nahe auch die Umformung des zugrundeliegenden Datenmodells über die Bereitstellung einer entsprechenden Konfiguration zu steuern. Um flexibel zu bleiben, sollte das Ergebnis aber kein reales Produkt sein, sondern vielmehr eine rein virtuelle Abbildung.
Um all diese Umformungen in Echtzeit durchführen zu können beschäftigte sich die MR früh mit der In-Memory Technologie, insbesondere mit SAP HANA. Mit Hilfe dieser Technologie war es möglich die vorher in einer entsprechenden Konfigurationsmatrix festgehaltenen Umformungen in Echtzeit durchzuführen und so die Ergebnisse einer späteren tatsächlichen Transformation der zugrundeliegenden Systeme in Teilaspekten vorwegzunehmen und einen Blick auf die Zukunft zu ermöglichen. Diese Virtuelle Transformation dient als Grundlage für verschiedenste Steuerungsmaßnahmen (z.B. ein Plan/Ist-Reporting).
Auch wenn dieses Vorgehen bereits einen Erfolg darstellt, so ist es für die Bedürfnisse einer digitalisierten Welt immer noch nicht ausreichend, denn bisher müssen die Änderungen an der Konfigurationsmatrix durch die IT vorgenommen werden. Dieser Prozess ist immer noch zu langwierig und fehleranfällig. Deshalb wurde im Rahmen dieser Arbeit ein Tool entwickelt, dass es dem Fachbereich erlaubt die Konfiguration selbständig zu ändern. Dadurch wurde der Prozess der Änderung verkürzt und durch ein verifiziertes Tool abgesichert.

Die Herausforderung bei der Erstellung des Tools war, dass sich viele Rahmenbedingungen ändern können und sich das Tool eigenständig auf diese Änderungen reagieren soll. Deshalb wurde im Tool extensiv die Mittel der dynamischen Programmierung im ABAP Umfeld ausgenutzt. Anders ausgedrückt konnten wir die dem Thema zugrundeliegende Komplexität nicht reduzieren, aber wir haben zumindest das Wissen zur Beherrschung dieser Komplexität aus den Köpfen der Mitarbeiter in ein lauffähiges Programm transferiert. Damit ist das Wissen der Mitarbeiter ist konserviert und die Komplexität an einer zentralen Stelle adressiert.

Die Maschinenfabrik Reinhausen GmbH

Die Maschinenfabrik Reinhausen GmbH mit Hauptsitz in Regensburg ist ein mittelständisches Familienunternehmen. Das Unternehmen MR beschäftigt weltweit 3.300 Mitarbeiter. Im Geschäftsjahr 2016 erwirtschaftete das Unternehmen einen Umsatz von 750 Millionen Euro. Das Kerngeschäft der MR ist die Regelung und Steuerung von Leistungstransformatoren. Auf diesem Gebiet ist die MR Weltmarktführer - 50 % des weltweiten Stroms läuft durch ihre Produkte. Neben der Herstellung von Verbundhohlisolatoren, der Bearbeitung von Kunststoffzylindern beinhaltet das Produktportfolio der MR überwiegend Komplementärprodukte und zahlreiche Dienstleistungen rund um den Transformator [2]. Das macht das Unternehmen im Bereich Energietechnik zu einem kompetenten Ansprechpartner.

Bisherige Abbildung der strategischen Geschäftsfelder der MR mittels vorhandener Datenstrukturen

Um die einzelnen Strategisches Geschäftsfelder der MR steuern zu können ist ein entsprechendes Reporting notwendig. Dieses Reporting muss sich derzeit noch an den vorhanden Datenstrukturen orientieren. Dies sind vorallem Informationen über den genutzten Vertriebsweg, das verkaufte Produkt, die Unterscheidung nach Ersatzteil- und Produktgeschäft, die verkaufende Verkaufsorganisation und die Unterscheidung zwischen Produkt- und Projektgeschäft. Die Kombination aus diesen fünf Elementen bestimmt das SGF bei MR. D.h jede Kombination der möglichen Werte für diese fünf Merkmale wird einem SGF zugeordnet. Die so entsehende Matrix wurde bisher mittels MS Excel abgebildet. Diese Matrix unterstützt sowohl dem Fachbereich als auch der Geschäftsleitung bestmöglich bei der Marktbearbeitung – also dem Identifizieren potenzieller bzw. relevanter Märkte und wer zum Wettbewerbsumfeld gehört.
Die Matrix ermöglicht Umsatzauswertungen und COPA-Auswertung (Kosten, Deckungsbeiträge) nach SGF darzustellen..
Das bisherige Vorgehen zum erstellen der SGF Matrix war dabei bisher folgendes: Nach Angaben der Fachbereiche wurden folgenden Informationen erhoben:

Die SGF-Matrix wurde manuell mit Hilfe dem Microsoft Office Excel-Programm aufgestellt. Dies ist zeitaufwändig für die Mitarbeiter, die diese Matrix erstellen müssten.

Im Schnitt werden alle 2 - 3 Jahre neue Geschäftsfelder an der Matrix hinzugefügt. In 5 Jahren wurden im Schnitt 62 Änderungen durchgeführt. Bei jeder Änderung an der Matrix müssen wieder alle Informationen von Hand sowohl in der Matrix eingetragen werden als auch im Business Warehouse (BW)-System angepasst werden. Dieser Prozess war langwierig und fehleranfällig.

Die Matrix ist aufgrund neuer Geschäftsfelder mit der Zeit sehr komplex geworden. Um alle Informationen im Überblick behalten zu können, wurden Geschäftsfelder mit gemeinsamen Merkmalen in der Kopfzeile der Matrix gruppiert. Folglich verliert die Matrix an Ihrer Übersichtlichkeit, so dass sich die Auswertung der Umsätze und die Informationsgewinnung zu Kosten und Deckungsbeiträge anhand der in Microsoft Excel erstellten Matrix erschwert haben.

Die derzeitige Möglichkeit die SGF mittels Microsoft Excel Tabelle darzustellen und diese zu bearbeiten ist sehr komplex. Das verlangt zu viel Zeit und Ressourcen. Um aus dieser Komplexität rauszukommen, soll diese fehleranfälligen manuellen Tätigkeiten durch ein neues Berichtswesen-Tool ersetzt werden. Dafür wird ein Programm benötigt, das ein Cockpit – Benutzeroberfläche – implementiert und die Wünsche des Fachbereiches in das Cockpit bereitstellt.

Begriffsdefinitionen

In den folgenden Abschnitte können die ganze Syntax und alle Sprachelemente der Programmiersprache ABAP nicht erklärt, da das auch nicht möglich wäre. Stattdessen werden die wichtigsten Sprachelemente, die für die Implementierung des Cockpits nützlich sind, erläutert.

ABAP List Viewer

Laut [4] ist der *ABAP List Viewer* (ALV) ein Tool zur Anzeige von Massendaten, in Tabellen- und/oder Baumdarstellung. Der ALV bietet eine Reihe von interaktiven Standardfunktionen wie zum Beispiel: Drucken, Aufsteigend/Absteigend sortieren, Filter setzen, Exportieren und Grafik anzeigen. Diese Standardfunktionen können von den Entwicklern ausgeblendet, angepasst oder ergänzt werden, und brauchen nicht mehr implementiert zu werden. Zwei Programmiermodelle stehen zur Verfügung: das ALV Grid Control (verfügbar seit *SAP R/3 4.6*) und das ALVm Object Model (ab *SAP NetWeaver 2004*). Die Implementierung des Programmiermodells ALV Grid Control, welches in einen Container einzubetten ist, wird in dieser Bachelorarbeit ausgeführt.

Das zu implementierende Programmiermodell – ALV-Grid-Control – verwendet die Klasse *CL_GUI_ALV_GRID*. Diese Klasse hat eine Methode set-table-for-first-display, mit der die Listenausgabe formatiert wird und Daten an der Control übergeben werden. *Controls* sind Bereiche in einem Dynpro, die selbst als Grundlage für die Visualisierung des ALV-Grids dienen.
Die Methode *set_table_for_first_display* bietet drei wesentlichen Parameters:

- zu einem die Ausgabelisten bzw. die Ergebnisliste, die an den Parameter it_outtab übergeben werden,
- zu anderen ein Feldkatalog, der an den Parameter it_feldcatalogue übergeben wird,
- und eine Layout–Struktur – Voreinstellungen für die ALV-Ausgabe– , die an der Parameter is_layout übergeben wird.

Feldkatalog

Der Feldkatalog ist eine interne Tabelle mit Informationen über die darzustellenden Felder. Mit Hilfe dieser Tabelle wird zum Beispiel festgelegt, welche Felder beziehungsweise Spalten auf dem ALV-Grid anzuzeigen sind, welche Datentypen die Felder haben etc. Spalteneigenschaften der auszugebenen Liste kann über Felder des Katalogs beeinflusst werden.

Selektionsbild

Ein Selektionsbild ist ein spezielles Dynpro (Dynamisches Programm), das ohne Verwendung des Screen Painters mit den Anweisungen *SELECTION–SCREEN*, *PARAMETERS* und *SELECT–OPTIONS* im globalen Deklarationsteil von ausführbaren Programmen definiert werden kann [5]. Mit Selektionsbildern wird dem Anwender eine Eingabemaske für Wertemengen zur Verfügung gestellt, die zur Einschränkung bzw. Abgrenzung der Datenmenge genutzt wird, die von der Datenbank gelesen werden muss.

Dynpro

Dynpros sind frei definierbare Objekte, bei denen ABAP-Programmierer oder Benutzer mit Hilfe von Ein– und Ausgabefeldern, Listen usw. Informationen anzeigen oder eingeben können. Dynpros sind eine Form des Dialogs zwischen dem Benutzer und dem ABAP-Programm. Ein Dynpro wird im Programm durch eine eindeutige viertstellige Kennung – Dynpronummer genannt – identifiziert. Das Dynpro ermöglicht es dem Benutzer, Daten zu erfassen und anzuzeigen. Auf dem Dynpro können mit Hilfe von Drucktasten, Tabstrips, Table Controls und weiteren graphischen Elementen einen benutzerfreundlichen Dialog programmiert werden. Das Dynpro dient hauptsächlich als Träger weiterer Bildelemente [6]. Mit dem Screen Painter – Werkzeug der ABAP Workbench – können Dynpros zu einer Transaktion angelegt werden und mit Hilfe

von Bedien– und Anzeigeelementen definiert und gestaltet werden.
Ein Dynpro besteht aus einem Bildschirmbild und der zugrundeliegenden Ablauflogik. Die Ablauflogik ist ein Programm, das die Verarbeitung des Bildschirmbilds steuert. Die Ablauflogik eines Dynpro ist zumindest in zwei Ereignisblöcke aufgeteilt: Process Before Output (PBO) und Process After Input (PAI). Die Ereignisse PBO und PAI werden iterativ ausgelöst, d.h. nachdem das Ereignis PAI ausgelöst und die Funktion ausgeführt wurde, wird danach wieder automatisch das Ereignis PBO ausgelöst, so dass wieder der Programmcode in PBO ausgeführt wird.

Process Before Output (PBO)

Der Verarbeitungsblock zum PBO–Ereignis ist die Stelle, die abgearbeitet wird, bevor das Dynpro aufgerufen wird, bzw. auf dem Bildschirm erscheint. Der PBO–Verarbeitungsblock dient der Vorbereitung und Initialisierung von Programmvariablen und Bildschirmfeldern.

Process After Input (PAI)

Der Verarbeitungsblock zum PAI-Ereignis wird nach der Bearbeitung des Bildschirmbildes aufgerufen, um auf Eingabe – wie etwa das Klicken einer Drucktaste oder eines Buttons – zu reagieren. Der PAI-Verarbeitungsblock dient der Verarbeitung der Benutzereingaben. Jeder Benutzeraktion auf dem Bildschirm ist mit einem Funktionscode verknüpft. In die nachfolgende Grafik, wird der Zusammenhang zwischen Dynpro, PBO Und PAI aufgezeigt.

Funktionscode

Ein Funktionscode wird generell durch Betätigung eines Bedienelementes (wie z.B.: Schaltflache, Drucktaste) ausgelöst. Jede entsprechende Aktion auf dem Dynpro muss in seinen Attributen einen eindeutigen Funktionscode haben. Wenn ein Anwender auf die Drucktaste klickt, wird dies dem ABAP-Programm durch die Bereitstellung des Funktionscodes im Systemfeld syucomm signalisiert. Dies kann im Ereignisblock User-Command abgefragt werden und behandelt werden.

Container

Im ABAP.Programm wird der im Dynpro-Editor angelegte Bereich auf der Maske von einem so genannten Container verwaltet. Container sind spezielle Klasse im R/3System, deren Aufgabe ausschließlich darin liegt, derartige Bilschirmbereiche zu verwalten. Controls (Zum Beispiel: ALV-Grid, ALV-Tree etc.), die im System verwendet werden können, müssen über Container verwaltet werden. Da Container selbst auch Controls sind, ist es möglich, Container in Container einzufügen und somit den Custom–Bereich logisch zu unterteilen. [7].
Im SAP werden fünf verschieden Typen von Containern unterschieden, die jeweils durch eine ABAP–Klasse repräsentiertwerden. Die Klasse CL_GUI_CUSTOM_CONTAINER, welche in vorliegende Bachelorarbeit verwendet wird, verwaltet einen Custom–Control–Bereich auf einem Dynpro.
Bei der Gestaltung des Cockpits sollen Selektionsbild und ALV–Grid in einem selben Dynpro unterbringen. Diese Eigenschaft des zu implementierenden Cockpits ist auf Komfort der Benutzer gedacht. Hierfür kommen zwei Techniken im Einsatz: Container Control und Subscreen. *Subscreens* sind Bereiche auf dem Bildschirm, in denen andere Dynpros eingebettet werden können.
Container Control sind rechteckige Flächen auf dem Bildschirmbild eines Dynpros, die mit Namen versehen werden. Selektionsbilder sollen erst als Subscreen definiert werden und dann an ein Dynpro eingebunden werden. Das ALV–Grid soll mit Hilfe des Container Controls an dasselbe Dynpro gebunden werden.

Alle Änderungen in der Ergebnisliste sollen durch den Fachbereich ohne den IT-Eingriff erfolgen. Dafür muss die Ergebnisliste aus dem ALV-Grid zur Laufzeit generiert werden bzw. dynamisch gestaltet werden. Die Programmiersprache ABAP bietet schon Techniken zur dynamischen Programmierung, die diese Anforderung erfüllen. Datenreferenzen, Feldsymbole in Verbindung mit generischen Datentypen spielen eine große Rolle bei der dynamischen Programmierung.

Feldsymbole

Ein Feldsymbol ist ein symbolischer Name für ein Datenobjekt, dem zur Programmlaufzeit ein konkreter Speicherbereich zugewiesen werden kann. Über ein Feldsymbol wird auf ein zugewiesenes Datenobjekt durchgegriffen, das heißt alle Zugriffe werden mit dem zugewiesenen Datenobjekt durchgeführt [8]. Mit der Anweisung *FIELD-SYMBOLS* wird ein Feldsymbol vereinbart.

Datenreferenzen

Datenreferenzen sind Adressen von Datenobjekten. Um auf den Inhalt eines Datenobjekts zugreifen zu können, auf das eine Datenreferenz verweist, muss dieses zuerst dereferenziert werden. Die Dereferenzierung erfolgt über den Dereferenzierungsoperator (–>*): ASSIGN dref –>* TO <fs>. Diese Anweisung weist ein Feldsymbol <fs> das Datenobjekt zu, auf das die Datenreferenz in der Referenzvariable dref zeigt.

Interne Tabellen

Interne Tabellen bieten in ABAP die Funktionalität von Arrays. Ein wichtiges Einsatzgebiet ist z.B. die pro-

gramminterne Speicherung und Aufbereitung von Inhalten aus Datenbanktabellen. [9]. Interne Tabellen werden zeilenorientiert verarbeitet. Dabei werden die Daten über einer Schleife (LOOP…ENDLOOP) zeilenweise im Speicher abgelegt, wobei jede Zeile der internen Tabelle die gleiche Struktur bzw. den gleichen Arbeitsbereich hat.

In dieser Arbeit sollen weiterhin Funktionen implementiert werden, die dem Fachbereich die Möglichkeit geben, die Ergebnisliste zu manipulieren. Zudem werden Sprachelemente und Syntax zur Datenbankänderungen Programmierung verwendet. Für das Programmieren von Datenbankänderungen stehen dem Entwickler im ABAP die Befehle des Open SQL zur Verfügung: INSERT (Einfügen), UPDATE (Aktualisieren), MODIFY (Ändern), DELETE (Löschen).

Anforderungsdefinitionen

Um die Anforderungen an das zu implementierende SGF-Cockpit zu erfassen, wurden vor Ort Workshops mit dem Fachbereich durchgeführt. In diesen Workshops wurden die Wünsche an das zukünftigen Berichtswesen-Tool bzw. Cockpit erfasst. Daraus ergeben sich folgenden Anforderungen.

Das Cockpit soll:

- dem Anwender die Möglichkeit geben, die SGF in einer übersichtlichen und strukturieten Form ansehen zu können. Dafür soll ein ALV-Grid-Control implementiert werden.
- dem Anwender die Möglichkeit geben, die im ALV-Grid bestehenden Daten abfragen oder filtern zu können. Hierfür soll das SGF-Cockpit mit einem Selektionsbildbereich ausgestaltet werden.
- Dem Anwender die Möglichkeit geben, Einträge bzw. Zelleninhalte der Ergebnisliste ändern und speichern zu können. Dafür soll in der Ergebnisliste, die entsprechenden Zellen mit Dropdown-Listbox für das Auswählen der Daten ausgerichtet werden. Weiterhin soll das Diskette-Symbol – zum Sichern des geänderten Eintrags – in der Symbolleiste des Einstiegsdynpros aktiviert werden.
- Dem Anwender die Möglichkeit geben, die Ergebnisliste erweitern oder löschen zu können. Zudem sollen zwei weiteren Dynpros in das Einstiegsdynpro eingebettet werden. Diese Dynpros sollten dem Fachbereich Eingabefelder und Schaltflächen/Buttons für das Einfügen und Löschen von Datensätze bzw. Datenfelder in der Ergebnisliste anbieten.

Die durchzuführende Änderungen sollen nur auf der Ergebnisliste und nicht auf der Datenbank erfolgen!

Um die Akzeptanz des Cockpits gegenüber dem Anwender zu erhöhen, soll das zu implementierende SGF-Cockpit weiterhin folgenden Qualitäten-Anforderungen erfüllen:

Das Cockpit soll erweiterbar, zuverlässig und benutzbar sein.

Die nachstehenden Use-Case-Diagramm fasst das Cockpit in seinen Funktionalitäten zusammen.

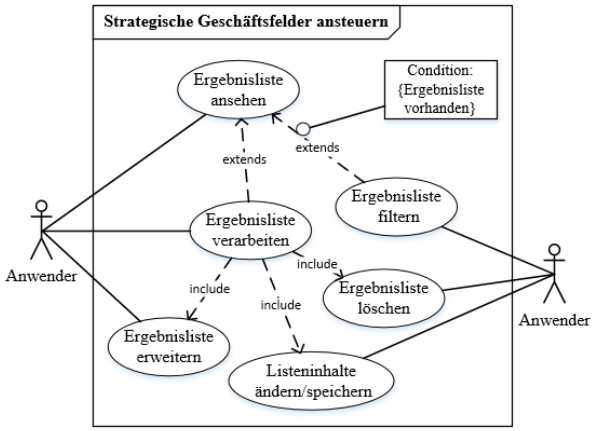

Abbildung 1: Use-Case Diagramm zum Ansteuern der SGF.

Um die Kommunikation bzw. Interaktion zwischen dem SGF-Cockpit und dem Fachbereich zu ermöglichen, werden Dynpros benötigt. Dafür müssen diese vorher zu einem Programm definiert werden. Dabei ist neben dem Layout bzw. Bildschirmmaske auch die sogenannte Ablauflogik des Dynpros zu realisieren.

- Die Layouts der anzulegenden Dynpros werden in der Entwurfsphase dieses Artikels vorgenommen.
- Die Ablauflogiken der entworfenen Dynpros werden in der Implementierungsphase realisiert.

Entwurfsphase

Ziel des Entwurfs ist es, die definierten Anforderungen an das Cockpit in eine „bildlich sichtbare“ [Benutzeroberfläche umzusetzen. In diesem Kapitel wird der Aufbau des Cockpits in seinen wesentlichen Elementen sowie dessen Architektur beschrieben und anhand diversen Screenshots ein erster Eindruck des SGF–Cockpits vermittelt. Zu Beginn werden die in diesem Projekt zu verwendenden Tabellen vorgestellt.

Benötige Datentabelle

Ein Ziel dieser Arbeit ist die SGF in Form eines ALV-Grid-Controls darzustellen. In dem ALV-Grid-Control müssen Daten aus drei Datenbanktabellen angezeigt werden. Diese sind: ZSGF_DATA, T179t und T25A7. Diese Datenbanktabellen sind alle miteinander über Fremdschlüssel verbunden.

Die wichtigste Datenbanktabelle ist ZSGF_DATA. Anhand dieser sollen Hauptdaten für die Darstellung des ALV-Grid-Controls beschafft werden. Bei der Transformation bzw. Verarbeitung der Ergebnisliste soll auf diese Datenbanktabelle verwiesen werden. Alle Daten der Datenbanktabelle ZSGF_DATA sind mit SAP-Schlüsseln gespeichert. Bei der Implementierung des ALV-Grid-Controls sollen aber einigen Daten mit entsprechenden Namen angezeigt werden. Um den entsprechenden Text bzw. Name zu diesen SAP-Schlüsseln zu haben, wird auf die Tabelle T25A7 und T179t verwiesen.

Die Datenbanktabelle T179 enthält alle mögliche Produkthierarchie, die es in der MR gibt. Sie wird für die Prüfung von Benutzereingabe verwendet.

Allgemeiner Aufbau

Damit das SGF-Cockpit einen einheitlichen und übersichtlichen Aufbau besitzt, empfiehlt es sich, dies in drei Benutzeroberflächen bzw. Dynpros zu untermauern. Jede Benutzeroberfläche im SAP-System wird über Dynpros aus realisiert. Dafür soll für jede Benutzeroberfläche eine eigene Dynpro angelegt werden. Somit sollen folgenden Dynpros in diesem Kapitel designt werden:

- Dynpro_0100 – Einstiegsdynpro – für das Einstiegsbild des SGF-Cockpits,
- Dynpro_9001 für das Bearbeiten der Datensätze des ALV-Grid-Outputs
- Dynpro_9002 für das Bearbeiten der Datenfelder des ALV-Grid-Outputs.

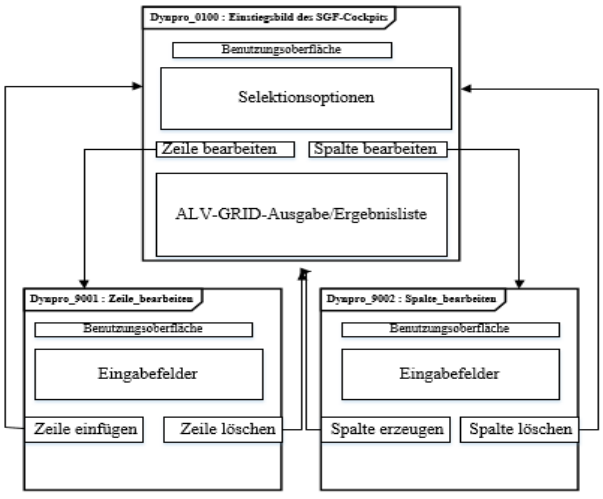

Abbildung 2: SGF-Cockpit-Architektur.

Die Abbildung 2 zeigt die Beziehung zwischen den Dynpros an. Die Pfeile Richtung verdeutlicht den Wechsel zwischen den jeweiligen Dynpros. z.B: klickt der Fachbereich auf die Taste “Zeile bearbeiten” in dem Dynpro_0100, wird er auf dem Dynpro_9001 gelandet und kann von dort aus einen Datensatz einfügen oder löschen. Das Verlassen eines Dynpro könnte auch über die GUI-Status erfolgen. Dafür müssen GUI-Status der jeweiligem Dynpro im Menü Painter festgelegt werden.

GUI-Status gestalten

Damit die dargestellten Dynpros verlassen werden können und die vom Benutzer geänderte Ergebnislisteninhalt gespeichert werden kann, sollte noch je Dynpro ein GUI-Status eingebaut werden, der bei Anzeige des Dynpros vorhanden ist. Dazu wird im PBO-Modul jedes Dynpro ein eigenes GUI-Status angelegt. In den Funktionstasten des Dynpro_0100 wurden nur nur die Speichern, Abbrechen, Zurück und Beenden Buttons aktiviert. Will das SGF-Cockpit auf diese Buttons reagieren so müssen diese Funktionen durch eigene Funktionscode belegt werden. Das ganze sollte dann folgendermaßen aussehen:

Abbildung 3: GUI-Status_0100: Funktionstastenleiste anlegen.

Einstiegsbild des SGF-Cockpits gestalten

Auf dem Dynpro_0100 (Vgl: Abb.2) sollen folgenden Bereiche reserviert werden: Ein Subscreen-Bereich für Selektionsbild, Drucktasten – Zeile bearbeiten und Spalte bearbeiten –, und ein Container-Bereich. Der Container-Bereich dient selbst als Grundlage für die Visualisierung des ALV-Grid-Controls.

Ansicht Spalte_bearbeiten

Die Ansicht Spalte_bearbeiten wird über das Dynpro_9002 gestaltet. Auf diesem Dynpro soll dem Fachbereich angeboten werden, Datenfelder zu der Ergebnisliste erweitern bzw. löschen zu können.

Um eine Spalte in der Ergebnisliste einzufügen oder zu löschen, werden folgenden Elemente benötigt:

- ein Eingabemaske für Spaltenangaben
- 2 Drucktasten: “Spalte einfügen” und “Spalte löschen”

Um die oben aufgelisteten Elementen in das Dynpro 9002 zu platzieren, muss in der Symbolleiste des Screen Painters auf der Drucktaste Dictionary/Programmfelder–Fenster doppelgeklickt werden.
Das Fenster Dict/Programmfelder wird geöffnet. Danach muss ein Tabellenname eingetragen werden und die Schaltfläche „*holen aus Dict“* betätigen. Die Felder der Tabellenname erscheinen in einer Liste. Die benöti-

ge Felder müssen markiert und per Drag&Drop in die leere Bildschirmmaske platziert werden.

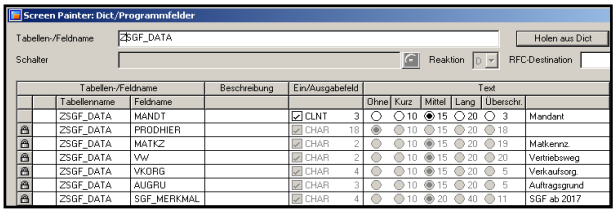

Abbildung 4: Auswahl von Eingabefeldern in das Dynpro 9002.

Das Platzieren von Drucktasten auf die Bearbeitungsfläche des Dynpro_9002 erfolgt über die Werkzeugleiste des Screen Painters. Dabei wichtig ist die Zuordnung von Funktionscoden an diesen Drucktasten. Somit können sie zum PAI-Ereignis und durch Benutzeraktion abgefangen und aufgelöst werden.
Die Drucktaste „Spalte einfügen“ wurde auf dem Dynpro mit dem Funktionscode „CREATE“ angelegt und wird zum PAI-Ereignis ementsprechen abgefragt. Die Drucktaste „Spalte löschen“ wird im Programm über den Funktionscode „DELETE“ identifiziert werden. Die folgende Graphik zeigt die Anordnung der Objekte des Dynpro_ 9002. Auf Komfort der Fachbereich wurde alle Objekte eingerahmt und betitelt.

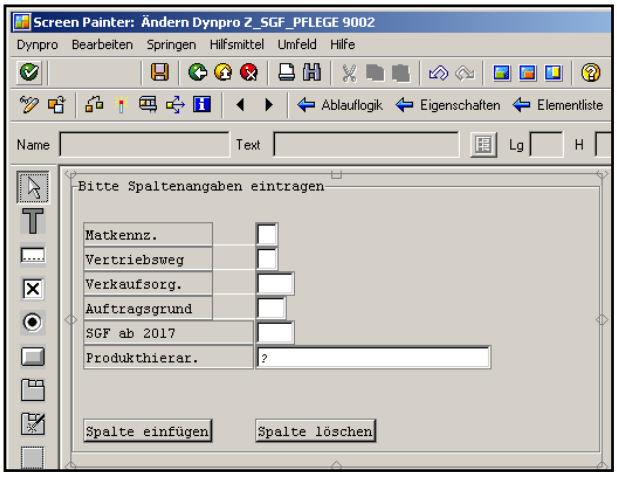

Abbildung 5: Platzieren von Eingabefelder und Drucktaste auf dem Dynpros 9002.

Die Ansicht Zeile_bearbeiten durch das Dynpro_9001 repräsentiert, wurde analog aufgebaut. Aber dort wird nur ein Eingabefeld “Produkthierarchie” benötigt. Da in der Datenbanktabelle ZSGF_ DATA wird eine Zeile durch das Feld Produkthierarchie eindeutig identifiziert.

Die Folgennde Abbildung zeigt das Ergebnis der Konfiguration des Ansichts Zeile_bearbeiten.

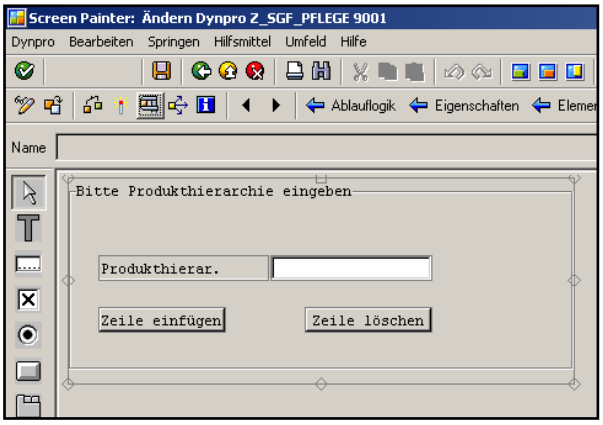

Abbildung 6: Konfiguration Ansicht Zeile_bearbeiten.

Nachdem die Layouts der Dynpros in der Entwurfphase realisiert wurden, werden in der Implementierungsphase die entsprechenden Ablauflogiken implementiert.

Implementierungsphase

Im folgenden Kapitel wird auf die Konkrete Umsetzung der definierte Anforderungen an das Cockpit eingegangen. Dabei wurden mehrere Form-Routinen auch Unterprogramme genannt implementiert. Für eine detaillierte Beschreibung der implementierung alle Form-Routinen wird auf die vollständige Bachelorarbeit verwiesen.

ALV-Grid-Control-Ausgabe

Zur ALV-Grid-Control-Ausgabe werden folgende Schritte durchlaufen:
und

Feldkatalog erstellen

Neben der Deklaration von Feldsymbole, interne Tabellen und Workareas – auch Struktur genannt – Typ für ALV-Ausgabetabelle, soll weiterhin ein Feldkatalog aufgebaut werden. Es wird bei dem Aufbau des Feldkataloges festgelegt, welche Spalten auf dem ALV anzuzeigen sind. Für die Erstellung des Feldkatalogs wurde eine Form-Routine Fieldcat_Build implementiert. Dabei erfolgte die Erstellung manuell. Der Feldkatalog wurde in Form einer Tabelle lt_fcat_temp vom Typ lvc_t_fcat aufgebaut und entsprechend der Wünsche von dem Fachbereich angepasst. Diese Tabelle wird später an die Methode *set_table_for_first_display* übergegeben

In der Abbildung 7 wird einen Codeabchnitt zum Feldkatalog Aufbau aufgezeigt.

```
*prodh TYPE c LENGTH 18,
 CLEAR LS_FIELDCAT.
 LS_FIELDCAT-COL_POS = 1.
 LS_FIELDCAT-FIELDNAME     = 'PRODH'.
 LS_FIELDCAT-KEY_SEL = 'X'.
 LS_FIELDCAT-EMPHASIZE = 'C500'.
 LS_FIELDCAT-DATATYPE = 'C40'.
 LS_FIELDCAT-OUTPUTLEN = 30.
 LS_FIELDCAT-FIX_COLUMN = 'X'.

 MOVE 'Produkthierarchie'(001) TO:
  LS_FIELDCAT-COLTEXT,
  LS_FIELDCAT-REPTEXT,
  LS_FIELDCAT-SCRTEXT_M,
  LS_FIELDCAT-SCRTEXT_S,
  LS_FIELDCAT-SCRTEXT_L.
 LS_FIELDCAT-STYLE = 32.
 APPEND LS_FIELDCAT    TO    LT_FCAT_TEMP.
```

Abbildung 7: Abchnittscode Feldkatalog aufbau.

Diese Abbildung zeigt, welche Ausgabeoptionen bzw. Eigenschaften das Feld ‚PRODH' in der Ergebnisliste haben soll. z.B das Feld ‚PRODH' soll als erste Spalte in dem ALV-Grid dargestellt werden, soll fixiert werden und soll grün gefärbt gefärbt. Über die Zeile der Feldkatalog-Tabelle wurden weitere Felder der Ergebnisliste beschrieben.

Weiterhin wurde über eine Layout-Struktur weitere Einstellungen für das Layout des ALV-Grid vorgenommen. Dabei wurden Feldnamen des Ausgabefelds für die Farben von Zeilen und Zellen, den Selketionsmodus und für die Optimierung der Spaltenbreite gesetzt.

Dynamischer Aufbau der Ergebnisliste

Es soll in dieser Arbeit dem Fachbereich die Möglichkeit geben, die Ergebnisliste aus dem ALV-Grid zu manipulieren. Hierfür ist es notwendig und sinvoll diese Liste zur Laufzeit zu generieren. Dafür wurde die Methode *cl_alv_table_create=>create_dynamic_table* verwendet (siehe Abb.Abbildung 8). Mit ihre wurde anhand des erstellten Feldkataloges die Ausgabeliste bzw. die Ergebnisliste, die später an den Parameter it_outtab der Methode *set_table_for_first_display* übergeben werden, aufgebaut. Die Bearbeitung dieser dynamischen erzeugten Tabelle erfolgt nicht mehr über Spaltenname sondern über Feldsymbole. Die einzelnen Felder des Arbeitsbereiches <dyn_wa> dieser Tabelle <gt_data> werden über den Namen der Felder mittels ASSIGN COMPONENT-Anweisung, angesprochen.

```
Create dynamic table
CALL METHOD CL_ALV_TABLE_CREATE=>CREATE_DYNAMIC_TABLE
  EXPORTING
    I_STYLE_TABLE             = 'X'
    IT_FIELDCATALOG           = LT_FCAT_TEMP
    I_LENGTH_IN_BYTE          =
  IMPORTING
    EP_TABLE                  = GT_DYN_ITAB.
ASSIGN GT_DYN_ITAB->* TO <GT_DATA>.
Workarea for the dyn. table
CREATE DATA D_LINE LIKE LINE OF <GT_DATA>.
ASSIGN D_LINE->* TO <DYN_WA>.
```

Abbildung 8: Dynamische interne Tabelle erzeugen.

ALV-Grid aufbauen

Im PBO-Modul des Dynpro_0100 wurden Referenzvariablen für das ALV Grid Control und den Custom_Container deklariert und instanziiert. Die Instanzierung erfolgte über das CREATE OBJECT-Befehl. Über den Exporting Parameter *parent* wurde das Container Control als Vater des ALV-Grid festgelegt. Dabei wird das Container Control an das Dynpro über den im Screen Painter angelegten Container gebunden.

Nachdem alle Einstellungen für die Ausgabe des ALV-Grid-Controls vorgenommen wurde, wurden die Ausgabetabelle bzw. die Ergebnisliste (gt_data) und die Strukturdaten (Layout und Feldkatalog) an das ALV-Grid über die Methode *set_table_for_first_display* übergegeben.

1. Selektionsbild aufbauen

Selektionsbild soll dem Benutzer erlauben, Parameters bzw. Selektionskriterien für die Datenselektion in der Ergebnisliste anzugeben. Dabei soll beachtet werden, dass Selektionsbild als Subscreen (Teildynpro) definiert werden. Abbildung 9 zeigt die Definition der Selektionskriterien im Selektionsbild. Diese Definition erfolgt über die SELECT-OPTIONS-Anweisung.

```
* Custom Selection Screen 1010
SELECTION-SCREEN BEGIN OF SCREEN 1010 AS SUBSCREEN.
SELECTION-SCREEN BEGIN OF BLOCK B1 WITH FRAME TITLE TEXT-004.
SELECT-OPTIONS   S_PRODH FOR ZSGF_DATA-PRODHIER .
SELECT-OPTIONS   S_MATKZ FOR ZSGF_DATA-MATKZ  .
SELECT-OPTIONS   S_VW FOR ZSGF_DATA-VW.
SELECT-OPTIONS   S_VKORG FOR ZSGF_DATA-VKORG.
SELECT-OPTIONS   S_AUGRU FOR ZSGF_DATA-AUGRU.
SELECTION-SCREEN END OF BLOCK B1.
SELECTION-SCREEN END OF SCREEN 1010.
```

Abbildung 9: Aufbau eines Selektionsbildes.

Wenn ein als Subscreen definiertes Selektionsbild in einem Dynpro **(**Dynamisches Programm**)** eingebunden werden soll, ist darauf zu achten, dass in der Ablauflogik des entsprechenden Dynpros sowohl zu PBO als auch zu PAI die Anweisung CALL SUBSCREEN ausgeführt werden muss, um die Daten zwischen Selektionsbild und ABAP-Programm zu transportieren.

2. Inhalte der Ergebnisliste ändern und Speichern

Inhalte der Ergebnisliste ändern

Um den Fachbereich das Ändern eines Zelleninhalts bzw. eines Eintrags in der Ergebnisliste zu erleichtern, wurde Dropdown-Listbox erstellt. um aus einer ALV-Zelle eine Listbox zu machen, soll zuerst im Feldkatalog an die entsprechenden Spalte das Feld DRDN_FIELD zugewiesen. In unserem Fall wurde die Spalte "Merkmal" dem Feld DRDN_FIELD zugewiesen.

Weiterhin sollte diese Listbox befüllt werden. Dazu wurde in einer Form-Routine set_dropdown_table einen Arbeitsbereich ls_dropdown mit Bezug auf die Dictionary-Struktur lvc_s_drop und eine interne Tabelle lt_dropdown mit Bezug auf die Dictionary-Tabelle lvc_t_drop angelegt und die relevanten Felder wurden gefüllt, wie etwa ls_dropdown-handle = '1' und ls_dropdown-value = '0070-Antriebe mit ISM'.

Danach wurden diese Felder über der Arbeitsbereich der interne Tabelle lt_dropdown hinzugefügt, etwa `APPEND LS_DROPDOWN TO LT_DROPDOWN.`

Abschließend wurde die Dropdown-Value dem ALV-Grid übergegeben werden.

Erfasste Änderung Speichern

Nachdem der Anwender aus der Dropdownl-Listbox einen Wert ausgewählt hat, muss er die *Enter-Taste* betätigen. Somit wird das Programm mitgeteilt, dass Änderung an einer Zelle der Ergebnisliste vorgenommen wurde. Zum abfangen bzw. zur Behandlung diese Event wurde eine Form-Routine Register_Edit implementiert. Zudem wurde die Methode Register_Edit_Event aufgerufen. Diese Methode stellt sicher, dass den ausgewählten Wert aus der Listbox im Programm vorgemerkt wird, nachdem die Enter-Taste gedrückt wurde.

Durch das Drücken des Sichern-Ikons (Funktionscode &SAVE_DATA) auf dem Dynpro_0100 sollen die geänderten Daten auf die Ergebnisliste geschrieben werden. Im PAI-Modul User_Command_0100 des Dynpro_0100 wird dazu das Unterprogramm Save_Change aufgerufen. Dabei soll zuerst die modifizierte Zelle in dem ALV-Grid erhaltet bzw. identifiziert werden. Zudem erfolgt eine Schleife über die dynamische interne Tabelle <*gt_data*>, um die geänderte Zelle in der Ergebnisliste zu suchen. Wird die entsprechende Zelle gefunden, wird sie über eine Read Table-Anweisung ausgelesen, dann in einer Struktur gespeichert und weiter an einer interne Tabelle angehängt. Der modifizierte Zelleninhalt ist nun bekannt und kann über der Modify-Befehl in der Ergebnisliste gestellt bzw. gespeichert werden.

3. Datensatz einfügen und Datenfeld löschen

In der Entwurfphase wurde den Aufbau der Ansicht Spalte_bearbeiten über das Dynpro_9002 realisiert. Für die Reaktion von Benutzereingaben, muss ein PAI-Modul entwickelt werden (siehe Abbildung 10). In diesem Modul werden die definierten Drucktasten und Funktionen für die Symbolleiste des Dynpro_9002 abgefangen bzw. behandelt.

```
MODULE USER_COMMAND_9002 INPUT.
  SAVE_OK = OK_CODE_SPALTE.
  CLEAR OK_CODE_SPALTE.
  CASE SAVE_OK.
    WHEN 'CREATE'.
      PERFORM INSERT_NEW_COLUMN.
      IF GV_EINGABE_KORREKT = 'X'.
        LEAVE TO SCREEN 100.
      ENDIF.
    WHEN 'DELETE'.
      PERFORM DELETE_COLUMN.
      LEAVE TO SCREEN 100.
    WHEN '&F03' OR '&F15' OR '&F12'.
      LEAVE TO SCREEN 0.
   WHEN OTHERS.
  ENDCASE.
ENDMODULE.
```

Abbildung 10: ABAP-Verarbeitungslogik-Dynpro_9002.

Das Feld save_ok dient der Übernahme von Funktionscodes aus dem Dynprofeld.

Datensatz einfügen

Damit der Benutzer einen Datensatz einfügt, steht ihm auf der Ansicht Zeile_bearbeiten ein Eingabefeld für das Lesen der Produkthierarchie-Nummer und die Drucktaste „Zeile *einfügen"* zur Verfügung. Trägt der Benutzer einen Wert in das Feld Produkthierarchie ein und betätigt die Drucktaste „Zeile *einfügen"* ruft das Programm die Form-Routine Insert_New_Row auf. Dabei wird zuerst geprüft, ob das Eingabefeld wirklich besetzt ist? Weiterhin wird geprüft, ob die eingegebene Produkthierarchie-Nummer in der Tabelle T179 vorhanden ist. Anschließend wird in der Ergebnisliste gesucht, ob der eingetragene Wert schon vorhanden ist. Falls nein, wird die eingegebene Produkthierarchie-Nummer in der Ergebnisliste eingefügt. Der Wert der Produkthierarchie-Nummer muss in der Ergebnisliste einmalig sein, da er jeden Datensatz eindeutig kennzeichnet. Über dem Insert-Befehl wird eine Zeile in der Ergebnisliste eingefügt. Hierfür wurde im Programm vor Aufruf des Insert-Befehls eine Struktur wa_data, die Daten der Datentabelle ZSGF_DATA einliest, angelegt. Danach wurde in dieser Struktur die einzufügende Zeile gestellt. Nachdem die Zeile eingefügt wurde, wurde mit dem Update-Befehl die Tabelle ZSGF_DATA aktualisiert. Wurde die Zeile erfolgreich in der Ergebnisliste eingefügt, wird den Benutzer über ein Pop-Up-Fenster informiert und beim Betätigen des Feldes "OK" verlässt der Benutzer der Ansicht Zeile_bearbeiten und landet er auf dem Einstiegsbild des Cockpits.

Das Unterstehende Ablaufdiagramm fasst die Form-Routine Insert_New_Row zusammen:

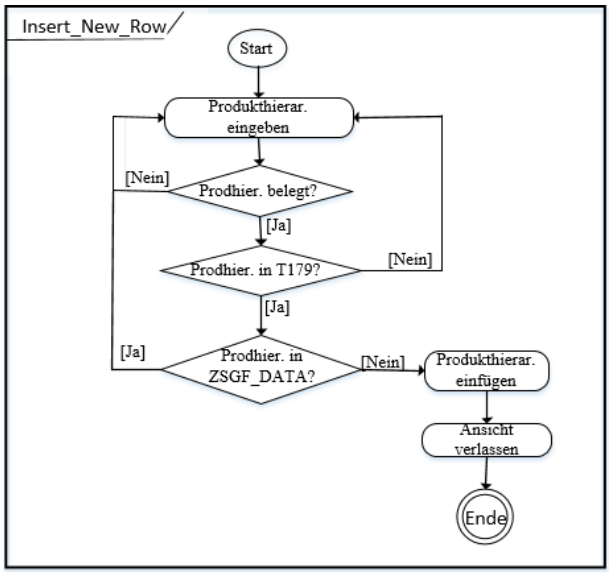

Abbildung 11: Ablaufsdiagramm Insert_new_Row.

Datenfeld löschen

Das Ereignis des Löschens eines Feldes wird durch Eingabe der zu löschenden Spalte und einen Klick auf die Schaltfläche „Spalte löschen“ im Bildschirmbild des Dynpro_9002 ausgelöst. Der dieser Schaltfläche zugeordnete Funktionscode „DELETE“ sorgt für den Aufruf der Form-Routine Delete_Column, welche wiederum eine Funktion „POPUP_TO_CONFIRM“ aufruft, um ein versehentliches Löschen einer Spalte oder eines Feldes zu verhindern. Dabei wird der Benutzer aufgefordert, seine gewünschte Aktion zu bestätigen. Drückt der Benutzer auf den *Yes*-Button, wird der Delete-Befehl ausgeführt. Hierbei wurde die zu löschende Spaltenangabe über eine Where-Klausel spezifiziert. Anschließend wird der Benutzer benachrichtigt über den Verlauf seiner Aktion.

```
IF LV_ANSWER_2 EQ '1'.   "If deletion is permitted
  "Delete data records from database table
  DELETE FROM ZSGF_DATA
  WHERE
   VW     = ZSGF_DATA-VW AND
   MATKZ  = ZSGF_DATA-MATKZ AND
   VKORG  = ZSGF_DATA-VKORG AND
   AUGRU  = ZSGF_DATA-AUGRU AND
   SGF_MERKMAL = ZSGF_DATA-SGF_MERKMAL.

  CLEAR: WA_DATA, ZSGF_DATA.
  MESSAGE 'Column has been removed' TYPE 'I'.
  LEAVE TO SCREEN 100.
ELSEIF   "If deletion is not permitted
  LV_ANSWER_2 EQ '2'.
  MESSAGE 'Column has not been removed' TYPE 'I'.
  LEAVE TO SCREEN 100.
ENDIF.
```

Abbildung 12: Datenfeld aus der Ergebnisliste löschen

Dynpro_9002 verlassen

Bei Betätigung der Cancel, Exit und Back Buttons auf der Symbolleiste wird das Dynpro_9002 verlassen und dem Anwender wird über die Anweisung – leave to screen 0 – in dem Object-Navigator Transaktion se80 – weitergeleitet.

Das Einfügen von Datenfeld und das Löschen von Datensatz in der Ergebnisliste wurden analog realisisert. Lediglich die Eingabefelder zum Einlesen aus der Datentabelle ZSGF_DATA unterscheiden sich in ihrer Anzahl.

Test-und Integrationsphase

Um die Qualität und Funktionalität des Programms zu gewährleisten, wurden sowohl Tests vom Entwickler, als auch von dem Fachbereich durchgeführt. Es wurden sowohl die funktionalen, als auch die nicht-funktionalen Anforderungen überprüft.

Überprüfen der nicht-funktionalen Anforderungen

Als erstes wurde getestet, ob das SGF-Cockpit sich erweitern bzw. modifizieren lässt. Durch die dynamische Programmierung sind Änderungen durch den Fachbereich einfach und schnell realisierbar.
Dieser dynamische Aspekt des Programms fördert auch die Wartbarkeit und die Performance. Als zweiten wurde die Applikation auf ihre Zuverlässigkeit geprüft. Das Cockpit reagiert zuverlässig auf alle möglichen Benutzereingaben Zum Schluss wurde die Anwendung auf ihre Benutzerfreundlichkeit getestet. Die Oberfläche unterstützt den Benutzer bei seinen Aufgaben. Sie ist einfach komfortabel bedienbar, mit Pop-up-Fenster ausgerichtet und mit wenigen Klicks lässt sich durch die Oberfläche navigieren. Dies wirkt sich positiv auf die Zufriedenheit der Benutzer im Umgang mit dem Cockpit aus.

Überprüfen der funktionalen Anforderungen

Die SGF wurde in Form eines ALV-Grids dargestellt. Die stehen SGF in übersichtlicher Form in das neue Cockpit zur Verfügung.
Des Weiteren sollte das Programm dem Anwender die Möglichkeit geben, die in das Cockpit bestehende SGF abfragen oder filtern zu können. Diese Anforderung wurde erfüllt, in dem Eingabefeldeder in das Einstiegsbild des SGF-Cockpits zu Verfügung gestellt wurden. Diese Eingabefelder dienen zu Selektionsoptionen oder Suchkriterien. Es Wurde auch überprüft, ob bei nicht vorhandenen Selektionsoptionen eine leere Ergebnisliste ausgegeben wurde.
Eine weitere Anforderung an das Programm war die Änderungen von Ergebnislisteinhalte und die Speicherung diese Änderungen zu ermöglichen. Um dies gerecht zu werden, wurde eine Dropdown-Listbox erstellt. Mithilfe dieser Listbox ist der Anwender in der Lage, einen Wert für strategisches Geschäftsfeld-Merkmal aus der erstellten Listbox auszuwählen. Nach Auswahl des gewünschten Wertes muss diese aber durch die Betätigung der Enter-Tatse bestätigt werden. Somit merkt das Programm diese Auswahl vor, um der geänderte Inhalt nach Betätigung des Speicherbuttons bzw. das Diskette-

Symbol in der Symbolleiste in das Cockpit zu speichern.
Bei den anderen Steuerungsfunktionen, wie das Einfügen und Löschen von Datensätzen bzw. Datenfeldern, wurden überprüft ob diese ordnungsgemäß funktioniert.
Zum abschließenden Test des neuen Berichtswesen-Tools wurde diese den Fachbereich zum Abnahme_Test zur Verfügung gestellt. Dabei soll der Fachbereich selbst alle Funktionen ausführen und auf Fehlermeldungen achten. Falls Fehler auftreten, sollen diese schnellstmöglich dem Entwickler des Cockpits mitgeteilt werden.
Fehler sind bei Testen nicht aufgefallen. Somit sind alle Steuerungsfunktionen geprüft und das Cockpit zur Pflege der strategischen Geschäftsfelder kann nun in das Produktivsystem transportiert und wird über eine angelegte Dialogtransaktion für alle involvierten Fachbereich zur Verfügung gestellt.

Fazit

Das Ziel war die Implementierung eines Cockpits für die Visualisierung und Verarbeitung der Strategischen Geschäftsfelder der Maschinenfabrik Reinhausen GmbH.
Das hier in dieser Arbeit implementiertes Cockpit befindet sich auf dem Stand der Anforderungen des Fachbereiches und wurde eingesetzt. Somit konnte den Nutzungsumfang betreffenden Vorgaben erfüllt werden. Das Cockpit bietet eine benutzerfreundliche und übersichtliche Ergebnisliste der strategischen Geschäftsfelder, mit der der Fachbereich alle relevanten Informationen jederzeit im Überblick hat, um wichtige Entscheidungen zu treffen, Umsätze auszuwerten, Kosten und Deckungsbeiträge zu ermitteln. Durch das erstellte Customizing-Cockpit werden der Zugang, die Handhabung und die Verarbeitung der SGF-Merkmale vereinfacht. Da die Ergebnisliste dynamisch programmiert wurde, kann sie nun ohne den Eingriff des IT–Bereiches durch den Fachbereich direkt transformiert beziehungsweise verarbeitet werden. Dadurch werden die Änderungen schneller, auf einem einzigen System – SAP R/3– erfasst und die IT wird parallel entlastet. Das erspart Zeit und Ressourcen gegenüber der fehleranfälligen manuellen Verarbeitung der strategischen Geschäftsfelder. Außerdem wurde durch ausgeführte Tests eine höhere Datenqualität gewährleistet.
Für die heutige Nutzung ist der Funktionsumfang des SGF-Cockpits ausreichend. Dennoch ist das Programmieren einer Benutzeroberfläche oder eine Schnittelle niemals komplett fertig. Es bestehen zukünftig immer Möglichkeiten das Programmieren dieser Schnittelle noch zu verfeinern. Die Möglichkeiten sind dabei vom Ideenreichtum des Programmierers abhängig.

Literatur

[1] Peyman Azhari, Nilufar Faraby, Alexander Rossmann, Bernhard Steimel, Kai S. Wichmann. Digital Transformation Report 2014. Köln: neuland GmbH & Co. KG (2014) (siehe S. 3-16).

[2] Maschinenfabrik Reinhausen GmbH: Über MR. Zugriff am 12.06.2017. http://www.reinhausen.com/de/desktopdefault.aspx/tabid-1449/1774_read-4521/

[3] Walsh Gianfranco; Deseniss Alexander; Kilian Thomas: Marketing. Eine Einführung auf der Grundlage von Case Studies. 2. Auflage. Berlin: Springer Gabler 2013 (siehe S. 126)

[4] BC405 ABAP-Reports programmieren SAP NetWeaver. SAP AG 2006/Q2. (Seite 2-13)

[5] Köble, Josef . Entwicklung barrierefreier Software mit SAP NetWeaver®. 1. Auflage. Bonn: Galileo Press (SAP PRESS) 2007 (siehe S. 234)

[6] BC410 Entwicklung dynprobasierter Benutzerdialoge. *SAP AG 2006.* (Seite 10-13)

[7] Rainer Riekert. ABAP – Programmierung, Fortgeschrittene Programmiertechniken für ABAP. Addison-Wesley Verlag, München, 2001 (siehe S. 132)

[8] Keller, Horst / Jacobitz Joachim / Hochlehnert, Bernhard (Hg.): ABAP Objects Referenz.1. Auflage, Bonn: Galileo Press 2002.

[9] Horst Keller und Sascha Krüger. ABAPObjects: Einführung in die SAP-Programmierung. 2. Auflage. Bonn: Galileo Press 2005.