HOCHSCHULE FURTWANGEN UNIVERSITY
Wirtschafts-informatik

IT-gestützte Visualisierung globaler Wertschöpfungsketten für das Corporate Controlling

Thesis

zur Erlangung des Grades
Bachelor of Science
im Studiengang Wirtschaftsinformatik
an der Fakultät Wirtschaftsinformatik
der Hochschule Furtwangen University

vorgelegt von

Christian Völker

Referenten:	Prof. Dr. Bernadin Denzel Christian Decker (IM & C GmbH)
Eingereicht am:	29. Februar 2008

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Thesis „IT-gestützte Visualisierung globaler Wertschöpfungsketten für das Corporate Controlling“ selbständig und ohne unzulässige fremde Hilfe angefertigt habe.

Die verwendeten Quellen sind vollständig zitiert.

Furtwangen, den _____

_____
Christian Völker

Abstract

Immer mehr Konzerne weisen eine breiter werdende Wertschöpfungskette auf. Dies liegt an der steigenden Zahl der Unternehmenszusammenschlüsse, die aufgrund eines erhöhten Kostendrucks durch die Globalisierung zustande kommen. Die Group Costing & Profitablility Engine (GCP) liefert eine Applikation, die im Stande ist, eine konzernweite Wertschöpfungskette aufzubauen. Hierbei werden die globalen Herstellkosten für den Konzern errechnet und Zwischengewinne ausgewiesen, die durch konzerninterne Verkäufe angefallen sind. Bisher lag der Fokus der GCP auf der korrekten Errechnung des Konzernergebnisses, ohne die Struktur der jeweiligen Wertschöpfungskette genauer darzustellen. Die breiten Wertschöpfungsketten führen jedoch immer mehr dazu, den Überblick über die eigene Wertschöpfungsstruktur zu verlieren. Daher soll die GCP nun durch ein grafisches Werkzeug ergänzt werden, welches in der Lage ist, eine konzernweite Wertschöpfungskette, bzw. relevante Auszüge davon, darzustellen. Die vorliegende Arbeit beschäftigt sich mit der Entwicklung eines solchen Werkzeugs. Es wird dabei insbesondere auf die Anforderungen eingegangen, die verschiedene Anwender an eine solche Visualisierung haben. Einen weiteren Bestandteil bilden das Design und die Implementierung eines entsprechenden Prototyps. Die Besonderheit der Lösung besteht aus der Verwendung einer neuen Technik der SAP AG, die zur Anzeige von Netzwerken bestimmt ist. Diese Technik nennt sich JNET und wurde durch eine spezielle Methodik in SAP R/3 eingebunden. Die Arbeit zeigt als Ergebnis einen Prototyp der Visualisierung, der als gekapselte Funktion in mehrere GCP-Applikationen eingebunden wurde. Besonderheiten in der Wertschöpfungs-struktur, wie z. B. Rekursionen, können direkt aus der erstellten Visualisierung der Wertschöpfungskette abgelesen werden. Weiterhin besteht die Möglichkeit, durch die komplette Wertschöpfungskette zu navigieren.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ABAP	Advanced Business Application Programming
ALV	Advanced List Viewer
API	Application Programming Interface
APO	SAP Advanced Planner and Optimizer
BADI	Business Add In
BSP	Business Server Pages
BW	Business Information Warehouse
CI	GCP Costing Item
CRM	Customer Relationship Management
DDIC	Data Dictionary im SAP R/3
DOM	Document Object Model
Dynpro	Dynamisches Programm (engl. screen) in ABAP Kontext
ERP	Enterprise Resource Planning
FI	GCP Financial Item
GCP	Group Costing and Profitability Engine
GFW	SAP Business Graphics Framework
GUI	Graphical User Interface
HTML	Hypertext Markup Language
Java RE (JRE)	Java Runtime Environment
MVC	Model View Controller
PB	Precosting Balance
SAP	Früher: Systeme Anwendungen und Produkte in der Datenverarbeitung (Heute: Eigenname)
SCM	Supply Chain Management
SGML	Standard Generalized Markup Language
TTYPE	GCP Transaktionstyp
W3C	World Wide Web Consortium
WebAS	Web Application Server
XML	Extensible Markup Language

1 Einleitung

Im folgenden Abschnitt wird die Motivation zur vorliegenden Themenstellung erläutert. Weiterhin wird auf die aktuelle Ausgangssituation und die genaue Zielsetzung des Projekts eingegangen. Das Kapitel schließt mit dem Gesamtaufbau des Dokuments und einer Anleitung zum gezielten Lesen ab.

1.1 Motivation

Das Projekt wurde bei der Firma Informations Management & Consulting GmbH (IM&C) durchgeführt, die als Beratungsunternehmen in der Informationstechnologie-branche tätig ist. Der Fokus liegt hierbei im SAP Umfeld. Seit dem Jahr 2005 betreibt die IM&C die Group Costing and Profitability Engine (GCP) (vgl. dazu auch Kapitel 2.3). Im Rahmen dieser Softwarekomponente entstand die Aufgabenstellung der vorliegenden Arbeit. Die GCP ermöglicht weltweit agierenden Unternehmen eine konsolidierte Konzernergebnisrechnung. Hierbei ermittelt sie die globalen Herstellkosten und weist die angefallenen Zwischengewinne aus. Im Rahmen dieser Rechnungen, wird eine konzernweite Wertschöpfungskette aufgebaut, die bisher lediglich als Datenstruktur abgelegt und zur Berechnung herangezogen wurde. Immer mehr Unternehmen sind jetzt jedoch auch an der graphischen Struktur der globalen Wertschöpfungskette interessiert, da hier oftmals Verbesserungspotentiale schneller bzw. einfacher zu erkennen sind.

Bisher war die graphische Darstellung von Komponenten der Wertschöpfungskette nicht Bestandteil der GCP, was durch die vorliegende Arbeit ergänzt werden soll. Vorhandene Anwendungen sollen durch eine grafische Struktur ergänzt werden, die individuell vom Anwender eingestellt werden kann. Die graphische Anzeige soll eine leichtere Navigation durch die GCP Ergebnisse ermöglichen und somit die Benutzerfreundlichkeit steigern.

Weiterhin soll dieses Projekt als Vorläufer für eine grafisch gesteuerte Simulation von Szenarien innerhalb der GCP dienen. Bislang müssen die Anwender für jede Applikation einen Selektionsbildschirm ausfüllen bzw. Änderungen in der Struktur der Wertschöpfungskette manuell vornehmen. Simulationen sollen später automatisch mit den Werten und der Struktur des Wertschöpfungsgraphen durchgeführt werden können (vgl. dazu auch Kapitel 7.3).

1.2 Ausgangssituation

Die GCP liefert ihre Ergebnisse in die entsprechenden Datenbanktabellen eines SAP R/3 bzw. BW Systems. Hier findet sich ebenso die globale Wertschöpfungskette mit allen Knoten und Kanten. Die Aufbereitung von Ergebnissen der GCP beschränkt sich momentan auf zwei Programme. Dies ist einerseits der ALV (Advanced List Viewer) Browser, der es ermöglicht, eine Masse von Daten auszuwerten und zu vergleichen. Auf der anderen Seite kann man mit der GCP-Anzeigefunktion ein kalkuliertes Material bzw. Produkt in einem Baum darstellen und dessen Zugänge bzw. Abgänge auswerten.

Der ALV Browser bietet hierbei keine Möglichkeit zur Navigation durch die globale Wertschöpfungskette, da an dieser Stelle ohnehin meist Massendaten über mehrere

Produkte bzw. Werke angezeigt werden. Die Anzeigefunktion ermöglicht dagegen eine eingeschränkte Navigation über eine Stufe der Wertschöpfung, da man bei den konzerninternen Zugängen bzw. Abgängen durch Doppelklick in das Partnermaterial navigieren kann. Bestimmte Zusammenhänge und Strukturen sind hierbei jedoch nur schwer zu erkennen, da ein Material u. U. durch viele Bewegungen schnell unübersichtlich wird.

An dieser Stelle wird auf das Kapitel 3.2 verwiesen, welches die aktuelle Umgebung der Anwendungen vor dem Einbau des Navigationsfensters vorstellt.

1.3 Zielsetzung

Die Arbeit hat zum Ziel, eine Technik zu finden, die den Anforderungen an eine sinnvolle Visualisierung von Wertschöpfungsketten gewachsen ist und auch die künftig geplanten Erweiterungen unterstützt. Weiterhin soll ein Prototyp erstellt werden, der diese Technik verwendet und die in Kapitel 3 vorgestellten Anforderungen implementiert.

Das vorliegende Dokument soll dem Leser veranschaulichen, wie die vorgegebene Aufgabenstellung gelöst wurde. Alle in der Vorlaufzeit definierten Anforderungen sollten hierbei in eine Spezifikation überführt werden, wie sie später auch in vielen anderen Softwareprojekten anzutreffen ist. Neben der Spezifikation sollen das Design und die Realisierung des Prototyps erläutert werden.

1.4 Aufbau der Arbeit

In dieser Arbeit wird die Entwicklung einer Softwarekomponente zur IT-gestützten Visualisierung globaler Wertschöpfungsketten vorgestellt. Die Entwicklung umfasst hierbei die Analyse diverser Anforderungen, die Spezifikation und das Design der Lösung.

Im Allgemeinen ist das Dokument ähnlich eines Vorgehensmodells in der Softwareentwicklung aufgebaut. Dies lässt sich an der folgenden Abbildung illustrieren:

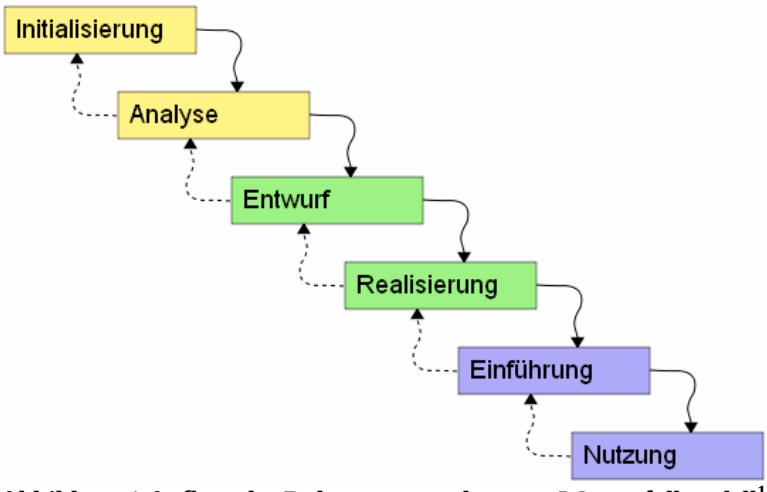


Abbildung 1 Aufbau des Dokuments analog zum Wasserfallmodell[1]

[1] Wikipedia Wasserfallmodell. In: http://upload.wikimedia.org/wikipedia/commons/e/e5/Wasserfallmodell.svg, zugegriffen am 12.11.2007.

Die Abbildung zeigt ein Wasserfallmodell mit Rücksprungmöglichkeiten. Hierbei kann man die einzelnen Phasen mit den Kapiteln der vorliegenden Arbeit verbinden. Die Initialisierung beschreibt in diesem Dokument die technischen und betriebswirtschaftlichen Grundlagen. Die Analyse wird durch die Kapitel Spezifikation und Anforderungen sowie die Auswahl der Technologien abgedeckt. Im weiteren Verlauf entsprechen sich Entwurf mit Design und die Realisierung mit dem gleichnamigen Kapitel in der Arbeit. Auf die Phasen Einführung und Nutzung wird in der Arbeit im Kapitel Zusammenfassung und Resultate bzw. Ausblick eingegangen. Der Leser kann in diesem Dokument die einzelnen Phasen des Softwareprojekts verfolgen.

Es folgt nun eine kurze Beschreibung aller Kapitel des Dokuments im Überblick. Das Dokument beginnt mit den notwendigen Grundlagen in Kapitel 2. Hierbei werden einerseits die verwendeten Techniken vorgestellt und andererseits Einblicke in die betriebswirtschaftlichen Bereiche gegeben. Den Schwerpunkt im Grundlagenkapitel bildet das GCP, in dessen Rahmen das gesamte Projekt durchgeführt wurde.

Im Kapitel 3 der Arbeit wird auf die verschiedenen funktionalen und nicht funktionalen Anforderungen eingegangen, die zusammen mit der Definition von Benutzerschnittstellen und Anwendungsfällen in eine Spezifikation überführt werden. Der Detailgrad der Spezifikation nimmt hierbei von oben nach unten zu. Es werden zunächst die allgemeinen Anforderungen definiert und diese später in einem Detailkapitel näher erläutert.

Das nächste Kapitel (Kapitel 4) beschäftigt sich mit der Auswahl von Technologien, um die Visualisierungskomponente den Anforderungen entsprechend zu realisieren. Hierbei werden zunächst alle Alternativen kurz vorgestellt. Im Anschluss daran wird die ausgewählte Technologie näher erläutert.

Nach der Festlegung auf eine Technologie, wird im darauf folgenden Kapitel 5 das Design der Software beschrieben. Auch in diesem Abschnitt liest man vom Allgemeinen ins Spezielle. Zunächst werden die Komponenten des Programms vorgestellt und die Klassen allgemein beschrieben. Im weiteren Verlauf erfolgen die Beschreibung der einzelnen Methoden und der Interaktion im Klassenmodell. Das Designkapitel vervollständigt sich mit der Vorstellung der Programmschnittstellen. Neben dem finalen Design finden sich in diesem Abschnitt auch einige Alternativen, wie die Programmkomponenten aufgebaut werden könnten. Darunter fallen auch die zugehörigen Laufzeitanalysen.

Das Kapitel Realisierung (Kapitel 6) beschreibt die einzelnen Phasen in der Entwicklung der Visualisierungskomponente. Hierbei wird insbesondere auf die Einteilung der Arbeitspakete und auf die Besonderheiten bzw. Probleme während der Realisierung eingegangen.

Den Abschluss der vorliegenden Arbeit bildet ein Resümee. Hier wird die gesamte Vorgehensweise des Projekts kritisch betrachtet und zusammengefasst. Weiterhin erfolgt eine Präsentation der erzielten Ergebnisse im Bezug zu den Anforderungen aus dem dritten Kapitel. Das Dokument schließt an dieser Stelle mit einem Ausblick ab, welches Potential dieses Projekt für künftige weitere Entwicklungen hat.

2 Grundlagen

Das folgende Kapitel soll zunächst die betriebswirtschaftlichen Grundlagen erläutern und einen Einblick in die verwendeten Techniken verschaffen. Es versteht sich somit als Glossar der Arbeit. Im Kapitel 2.3 wird insbesondere auf die GCP (Group Costing and Profitability Engine) eingegangen, die alle relevanten Daten für die grafische Anzeige bereitstellt.

2.1 Supply Chain Management

Die Basis des Supply Chain Managements bildet die Wertschöpfungskette (engl. supply chain). Diese ist als unternehmensübergreifendes Netzwerk zu verstehen, in dem ein Produkt von einem Lieferanten gekauft und später an den Kunden verkauft wird. Dabei entsteht unternehmensintern auf jeder Produktionsstufe ein Wertzuwachs (Wertschöpfung). Eine globale Wertschöpfungskette umfasst hierbei alle Unternehmen des Konzerns mit ihren unternehmensinternen Wertschöpfungsketten, die sich zu einem Netzwerk verbinden, von der „source of supply“ bis zum „point of consumption“.[2]

Die folgende Abbildung zeigt exemplarisch eine Wertschöpfungskette mit den Material-, Informations- und Geldflüssen.

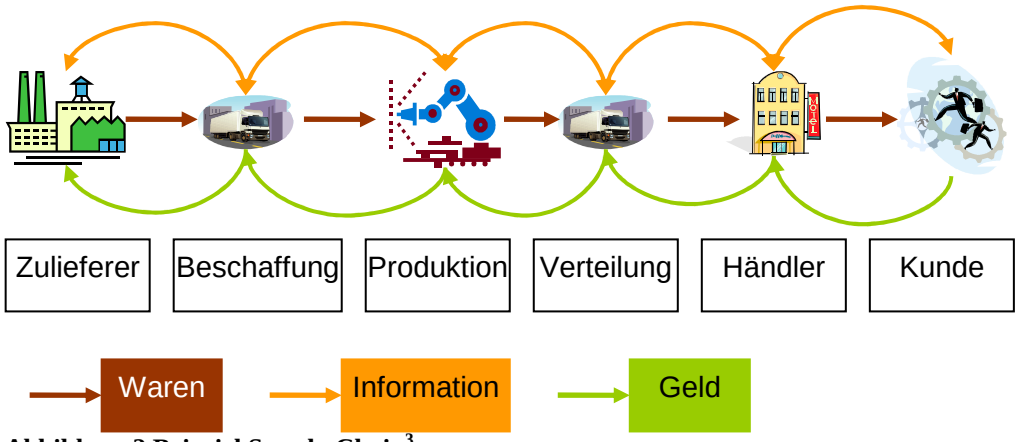


Abbildung 2 Beispiel Supply Chain[3]

Weiterhin werden die verschiedenen Stufen vom Zulieferer bis zum (End)kunden illustriert. Im Falle der globalen Wertschöpfungskette kann ein Konzern gleichzeitig Teilelieferant als auch Produzent ein. Der Materialfluss erfolgt immer in Richtung Endkunde. Der Informationsfluss erfolgt in beiden Richtungen, da Lieferant und Empfänger die Lieferung bzw. Bestellung abstimmen müssen. Der Geldfluss erfolgt vom Kunden zurück zum Händler usw. Innerhalb von Wertschöpfungsketten kann es zu einer Besonderheit kommen, die sich Rekursion nennt. Bei einem solchen Prozess fliest ein Material direkt oder indirekt wieder in sich selbst ein. Es entsteht also ein Zyklus, der mehrere Materialien umfassen kann. Diese Rekursionen müssen bei der Berechnung und der Anzeige globaler Wertschöpfungsketten unbedingt berücksichtigt werden. Als Beispiel soll hier ein Szenario in der Produktion eines Motorsägenherstellers dienen. Hier wird aus dem Rohstoff Magnesium ein Motorblock gegossen. Bei der

[2] Vgl. Busch, Axel (2004), S. 4.
[3] Vgl. Kuhn, Axel; Hellingrath Bernd (2002), S. 10.

Weiterverarbeitung wird in den Block gebohrt, wobei als Abfallprodukt Magnesiumspäne entstehen. Diese werden wieder eingeschmolzen und für den nächsten Motorblock verwendet.

Das SCM (Supply Chain Management) definieren Scholz-Reiter und Jakobza wie folgt.

> *„Supply Chain Management, auch Lieferkettenmanagement, ist die unternehmensübergreifende Koordination der Material- und Informationsflüsse über den gesamten Wertschöpfungsprozess von der Rohstoffgewinnung bis über die einzelnen Veredelungsstufen bis hin zum Endkunden mit dem Ziel, den Gesamtprozess sowohl kosten- als auch zeitoptimal zu gestalten.“*[4]

Das SCM koordiniert also alle Vorgänge und Aktivitäten an einem Produkt oder Material, vom Rohstoffeinkauf bis zum Verkauf an den Kunden. Gerade das Management globaler Wertschöpfungsketten wird in Zukunft eine immer höhere Bedeutung erhalten, da es vermehrt zu Unternehmenszusammenschlüssen kommt, und die Globalisierung stetig zunimmt.[5] Somit sehen sich viele produzierende Unternehmen aus dem Kostendruck gezwungen, die Produktion in neue Werke in Billiglohnländer zu verschieben. Hier steht das globale SCM einer neuen Herausforderung gegenüber.

2.2 SAP

Die Firma SAP wurde im Jahr 1992 in Mannheim von fünf ehemaligen Mitgliedern der Firma IBM gegründet[6]. Die SAP entwickelt branchenneutrale Standardsoftware im Bereich ERP (Enterprise Resource Planning), CRM (Customer Relationship Management) und SCM. Grundsätzlich ist die SAP Software für große Unternehmen gedacht, gerade in jüngster Zeit versucht man aber auch die kleinen bis mittelständischen Unternehmen anzusprechen. Die SAP stellte diesbezüglich in den vergangenen Jahren eine Reihe neuer Software vor z.B. MySAP Suite und Netweaver.

Das 1988 entwickelte SAP R/3 System ist eine ERP Standardlösung für mittlere und vor allem große Unternehmen. Es bezeichnet den Nachfolger des 1979 eingeführten R/2 Systems, wobei R für Real time steht, was eine umgehende Verarbeitung und Speicherung zentraler Daten bedeutet[7]. Während R/2 noch rein auf Großrechnern mit Terminals installiert war, wurde die Architektur des R/3 in mehrere Stufen eingeteilt. Es gibt einen Datenbankserver, mindestens einen Applikationsserver und den Präsentationsserver, der in Form des SAP GUI (Graphical User Interface) auf dem PC des Anwenders installiert ist. SAP R/3 ist in verschiedene Module eingeteilt, wobei man die drei Hauptbereiche Finanzwesen, Logistik und Personalwirtschaft abgrenzen kann. Die Besonderheit von R/3 ist hierbei die Integration der Module. Werden z.B. bestimmte Materialbuchungen vorgenommen, werden automatisch auch die zugehörigen Belege im Finanzwesen erzeugt und die Werte für das Controlling fortgeschrieben. Ein weiteres Beispiel bildet das Personalwesen, dessen Daten z.B. auch bei der Arbeitsplanung von Aufträgen in der Produktionslogistik verfügbar sind. Hierbei greifen alle Applikationen auf einen zentral gehaltenen Datenbestand zu. Das SAP R/3

[4] Scholz-Reiter, Bernd; Jakobza, Jens (1999), S. 7ff.
[5] Vgl. Kuhn, Axel; Hellingrath Bernd (2002), S. 37.
[6] Vgl. Computer Data Institut (Hrsg.) (2001), S. 19.
[7] Ebd., S. 20f.

System ist branchenunabhängig und somit in fast jedem Konzern einsetzbar. Zur kundenindividuellen Anpassung des Systems steht ein umfangreiches Customizing zur Verfügung, in welchem Vorlagen für die verschiedenen Branchen zur Verfügung stehen. Die branchenspezifischen Einstellungen können vom Kunden nun weiter spezialisiert werden.

Neben dem Customizing kann der Kunde sein System durch eigene Programme erweitern. Hierfür steht mit ABAP (Advanced Business Application Programming) eine eigene Programmiersprache mit einer umfangreichen API (Application Programming Interface) zur Verfügung.[8] Während die eigentlichen Kunden i. d. R. hierbei nur kleine Programmteile durch sog. Customer-Includes, User-Exits und BADIs (Business Add Ins) anpassen, können Beratungsunternehmen wie die IM&C eigene größere Entwicklungen in speziellen Namensräumen vornehmen, welcher durch einen speziellen Prefix vor dem Programmnamen abgebildet wird. Die User-Exits, Customer-Includes und BADIs sind hierbei vordefinierte Absprünge aus dem Standard, um kundenindividuelle Funktionalitäten zu implementieren. Hierbei wird auch sichergestellt, dass ein Wechsel des Release keine Auswirkungen auf den kundenspezifischen Programmcode hat.[9] Die Erweiterung in kundeneigenen Namensräumen, erlaubt es, eigene Applikationen zu entwickeln, die dabei vollkommen unabhängig zum Standard betrieben werden können.

Die Programmiersprache ABAP nennt sich im verwendeten Basis Release 4.6C bereits ABAP Objects und umfasst eine Vielzahl von Werkzeugen zur objektorientierten Entwicklung. Neben diesen Entwicklungswerkzeugen stellt die SAP auch einige Standardklassen z.B. für die XML (Extensible Markup Language) Verarbeitung und die grafischen Controls, wie Bäume und Listen, zur Verfügung. Die Sprache ABAP ist aber nicht als vollständig objektorientiert zu verstehen, da sie ebenso noch alle Merkmale einer prozeduralen Programmiersprache enthält.

In den neueren Basis Releases der R/3 Software entstanden im Rahmen einer Erweiterung um den WebAS (Web Application Server) weitere Internetkomponenten, wie beispielsweise die BSP (Business Server Pages), die SAPs Pendant zu den Java Server Pages darstellen. Die BSP erlauben es, Daten aus dem R/3 System in dynamisch erstellten Webseiten anzuzeigen. Hierbei stehen ABAP spezifische Kommandos in Form von Taglibs zur Verfügung, die der WebAS interpretieren kann.

Neben SAP R/3 kann GCP auch mit einem SAP BW (Business Warehouse) betrieben werden. Das SAP BW ist eine Data-Warehouse Lösung der SAP, die weitaus mehr Analysemöglichkeiten bietet als eine traditionelle Datenbank. Das BW bietet weiterhin einige Business Intelligence Werkzeuge an, um die Daten zu untersuchen und neue Erkenntnisse zu gewinnen[10]. Bis zu einer gewissen Schicht in der Software gleichen sich BW und R/3 in der Basis. Der Einsatz von BW entlastet die operativen R/3 Systeme, da aufwendige Analysen somit extern durchgeführt werden können.

[8] Vgl. Gadatsch, Andreas (2006), S. 10f.
[9] Vgl. https://www.sdn.sap.com/irj/sdn/wiki?path=/display/SRM/BAdI+-+general+information& eingesehen am 27.12.2007.
[10] Vgl. Chamoni, Peter; Gluchowski, Peter; Hahne Michael (2005), S. 37.

2.3 Group Costing and Profitability Engine (GCP)

In diesem Abschnitt soll die GCP Applikation vorgestellt werden, deren zentrale Anzeigefunktion durch eine grafische Navigation ergänzt werden soll. Es ist ebenso denkbar, weitere Funktionen der GCP mit einer grafischen Navigation auszustatten. Dazu werden weitere Projekte folgen. Die GCP ist ebenso von hoher Bedeutung, da sie die Daten und Ergebnisse liefert, aus denen eine grafische Darstellung der Wertschöpfungskette aufbereitet werden soll.

Die Globalisierung und der Trend zu immer mehr Unternehmenszusammenschlüssen führen zu immer breiter werdenden Wertschöpfungsketten in Konzernen. Die somit komplexer werdenden Strukturen erschweren eine gezielte Auswertung und Analyse des gesamten Ergebnisses. Die Systemlandschaft kann dabei nicht nur über die ganze Welt verteilt, sondern auch heterogen sein. SAP lässt an dieser Stelle auch keine Betrachtung der Wertschöpfungskette über die Systemgrenzen hinweg zu, was zu einer Bildung von Wertschöpfungsinseln führt.[11]

Vor der Entwicklung der GCP mussten Konzerne ihre Ergebnisse lokal berechnen und diese mit Hilfe von Excel oder kundenindividuellen Entwicklungen zusammentragen. Die Errechnung des globalen Ergebnisses auf Basis dieser Datensammlung führt hiermit nur zu sehr ungenauen Werten. Die GCP sorgt hier für Abhilfe. Diese Applikation ermöglicht eine integrierte Berechnung des Konzernergebnisses. Hierfür stellt die GCP Standardschnittstellen zu SAP und anderen Programmen, wie z.B. Excel, zur Verfügung. GCP ist ein Add-On für SAP R/3 und komplett in ABAP entwickelt. Die GCP ist in mehrere Programme aufgeteilt, die in einer bestimmten Reihenfolge ablaufen müssen und über ein eigenes Customizing verfügen.

Der Lösungsansatz besteht hierbei aus der Verbindung der Wertschöpfungsinseln und Zusammenführung von Mengen und Werten aus den unterschiedlichen Quellsystemen. In einem zentralen System werden diese Daten in einer einheitlichen Datenstruktur gesammelt und eine globale Kostenschichtung erstellt.[12]

Die GCP arbeitet in vier aufeinander folgenden Phasen, die in den nächsten Abschnitten nun beschrieben werden. Die folgende Abbildung stellt die Applikationsreihenfolge der GCP vor:

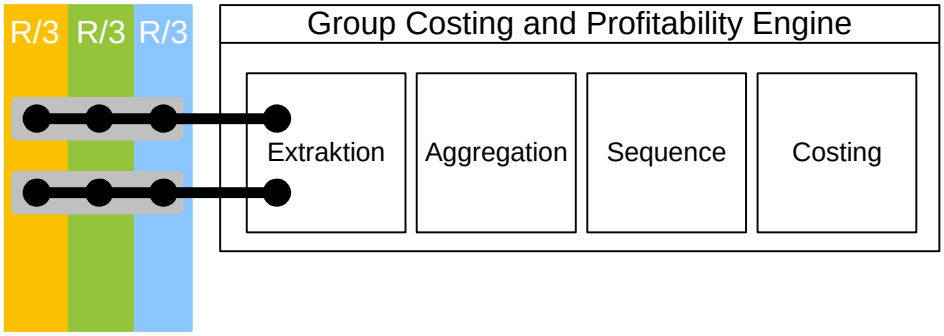


Abbildung 3 GCP Applikationsreihenfolge[13]

[11] Vgl. Wurm, Fritz (2005), S. 1.
[12] Quelle interner Vortrag.
[13] Ebd.

Die Extraktion holt die Daten aus den Quellsystemen (SAP R/3). Darauf folgen die Aggregation, die Sequence und schließlich das Costing.

2.3.1 Extraktion

Die Extraktion beschafft die GCP relevanten Daten in den verschiedenen Quellsystemen. Hierbei werden die Daten in eine einheitliche Struktur, die SI (Single Items), überführt und im zentralen System abgelegt. Die folgende Abbildung (s. Abbildung 4) stellt die Vorgehensweise der Extraktion schematisch dar.

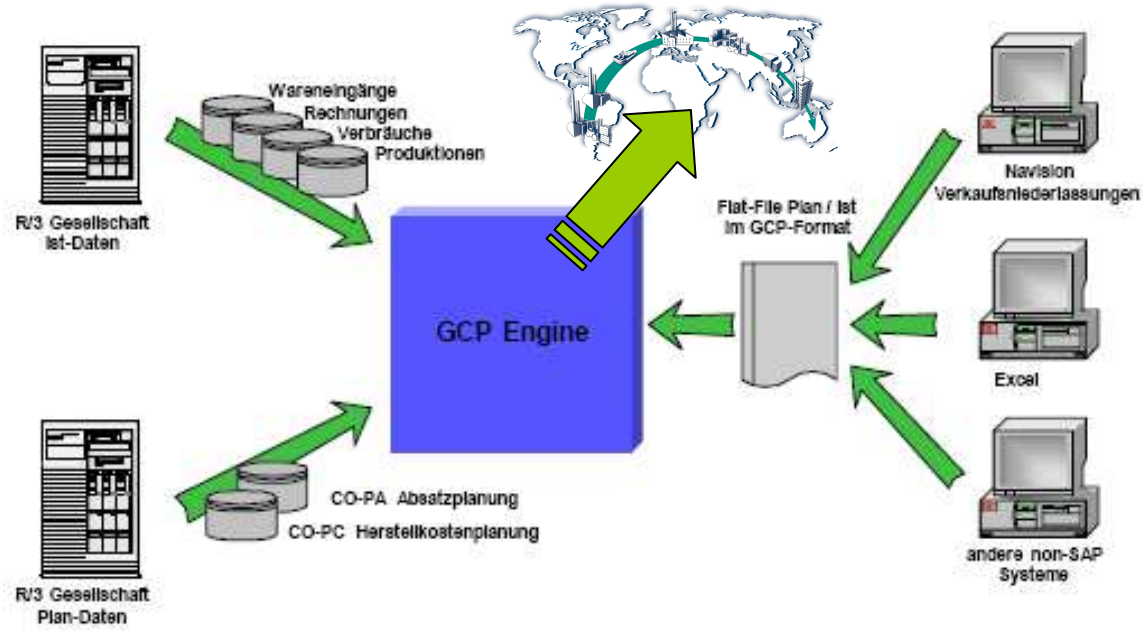


Abbildung 4 GCP Extraktion[14]

Die GCP kann sowohl Plan- als auch Istdaten aus den Quellsystemen extrahieren. Die Unterscheidung nach Plan und Ist erfolgt durch eine Trennung der Daten in Versionen. Neben der Extraktion aus SAP Systemen, können die Daten auch über eine Import-Schnittstelle z.B. als CSF oder Excel bereitgestellt werden. Die korrekte Extraktion der Quelldaten ist essentiell für die weiteren Verarbeitungsschritte. Das Ergebnis der Extraktion bilden diese sog. SI, in denen ein einzelner Vorgang abgebildet wird. Ein Vorgang ist hierbei als Buchung im SAP R/3 zu verstehen. Als Beispiel soll an dieser Stelle ein Wareneingangsbeleg aus der Materialwirtschaft (Modul MM) dienen. Hierbei wird aus jeder Buchung eines Wareneingangs in der Materialwirtschaft genau ein Single Item, das die Menge des Wareneingangs abbildet. Die Vorgänge werden durch Vorgangsarten, sog. Transaktionstypen unterschieden z.B. Wareneingang, Warenausgang, Umbuchung etc. Jede Bewegung, die für die Berechnung des globalen Ergebnisses relevant sein soll, muss als SI vorliegen. Es ist weiterhin darauf zu achten, dass in SAP mehrere Belege für einen Vorgang erzeugt werden, je nach dem welche SAP Module aktiv sind. Wird beispielsweise ein aktives FI-Modul betrieben, muss der o. g. Wareneingang auch buchhalterisch erfasst werden. Dies erfolgt in einem FI-Beleg, der ebenfalls von der Extraktion in ein separates SI überführt wird. Die GCP verarbeitet die Vorgänge also, ähnlich wie SAP, getrennt nach Mengen- und Werteflüssen.

[14] Quelle interner Vortrag.

2.3.2 Aggregation

Die Aggregation setzt auf den durch die Extraktion ermittelten SI auf und fasst hierbei gleichartige Vorgänge zusammen. Wird bspw. ein gleiches Produkt mehrmals vom gleichen Lieferanten eingekauft, können alle diese Beschaffungen in der Folgebetrachtung als Gesamtvorgang gesehen werden. Im Zuge dieser Gruppierung erfolgt zusätzlich eine Umrechnung der Werte in die lokale Firmenwährung, die historische Währung des Konzerns und die globale Währung. Die resultierenden, zusammengefassten Sätze werden FI (Financial Items) genannt und sind die Basis aller nachfolgenden Schritte. Entgegen des historisch entstandenen Namens umfassen die FI neben den Werteflüssen auch Materialflüsse.

Im Rahmen der schrittweisen Verarbeitung von SI wird in der Aggregation weiterhin der globale Wertschöpfungsgraph mit seinen Knoten und Kanten erzeugt. Dies erfolgt mit der Hilfe von Partnerinformationen aus den Single- bzw. Financial- Items. Die Partnerinformationen verweisen hierbei auf den jeweiligen Sender bzw. Empfänger des aktuellen Knotens. Aus betriebswirtschaftlicher Sicht stellt ein Knoten ein Material dar, welches in einem bestimmten Werk vorhanden ist. Ein solcher Knoten umfasst alle FI für dieses Material in diesem Werk und wird CI (Costing Item) genannt. Die folgende Grafik soll eine Abgrenzung der Datenstrukturen SI, FI und CI vornehmen:

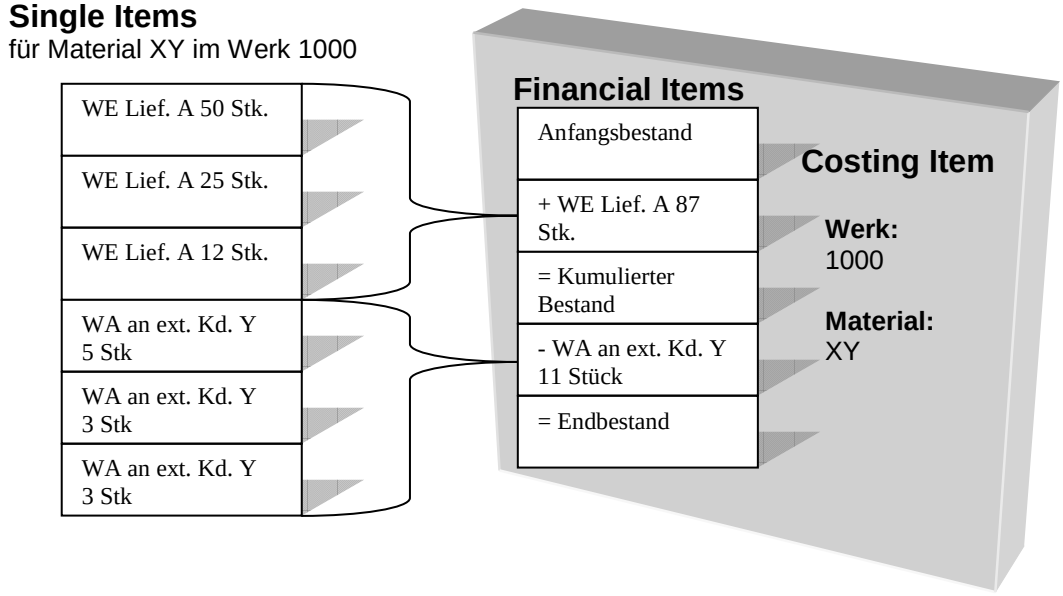


Abbildung 5 Abgrenzung GCP Datenstrukturen

Die Grafik zeigt auf der linken Seite eine Menge von Single Items, die von der Aggregation in zwei Financial Items zusammengefasst werden, da Lieferant bzw. Kunde übereinstimmen. Auf der rechten Seite ist ein großer Block dargestellt, der das Costing Item beschreibt, welchem die Financial Items zugeordnet sind. Alle SI und FI in der Abbildung beziehen sich also auf das Material „XY“ im Werk „1000“.

Die Kanten des Wertschöpfungsgraphen (s. Abbildung 6) zeigen die Beziehungen zwischen den Materialien und stellen somit Verbrauch, Einkäufe und Umlagerungen dar. Die Knoten in der Abbildung bilden die CI der Wertschöpfungskette ab, welche konkret ein Material in einem bestimmten Werk beschreiben.

Diese Kanten und Knoten werden separat auf der Datenbank abgelegt und dienen als Grundlage zur graphischen Darstellung der globalen Wertschöpfungskette. Die folgende Abbildung (s. Abbildung 6) zeigt den Wertschöpfungsgraphen nach der Aggregation.

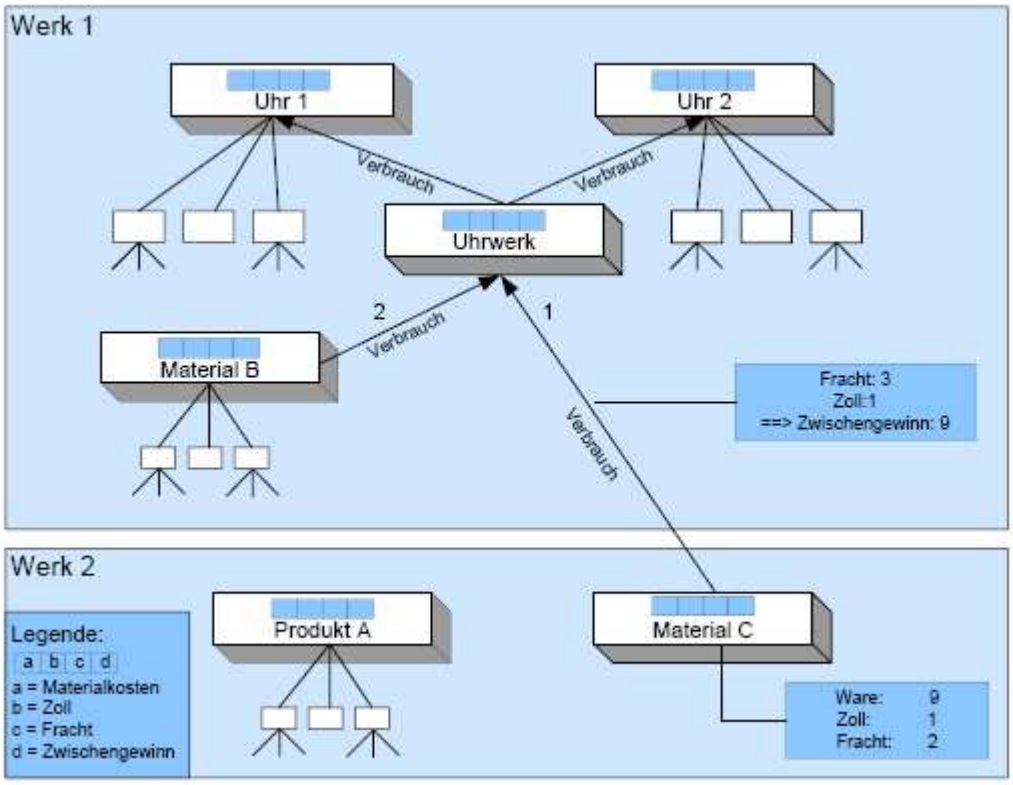


Abbildung 6 Wertschöpfungsgraph nach Aggregation[15]

In Werk 1 und Werk 2 sind die einzelnen Materialien mit ihren Beziehungen abgebildet. Die einzelnen Kostenelemente(s. Abbildung unten links) tragen aber noch keine Werte, da diese erst im Costing kalkuliert werden. Die Kanten tragen neben der Herkunfts- und Zielinformation auch die kumulierte Menge der Bewegungen über die Periode.

2.3.3 Sequence

Im Anschluss an die Aggregation übernimmt die Sequence die Reihenfolgebestimmung der Knoten des zuvor erzeugten Wertschöpfungsgraphen. Die Bestimmung der Reihenfolge wird dadurch notwendig, da die Knoten und Kanten vergleichbar einer Adjazenzmatrix vorliegen und somit keinerlei Rückschlüsse auf die „Position“ des Knotens innerhalb des Graphen zulassen. Eine Adjazenzmatrix beschreibt hierbei die relationale Darstellung eines Graphen. Für N Knoten wird hierbei eine N:N Matrix aufgebaut, um festzustellen, ob eine Verbindung zwischen zwei Knoten vorhanden ist.[16] Bei der Erstellung des Wertschöpfungsgraphs in der Aggregation ist die Kalkulationsstufe für das entsprechende CI (Knoten) noch nicht bekannt. Man kann lediglich eine Aussage über die Beziehung mit anderen Knoten treffen. Die Sequence muss daher bestimmen, wo die Anfangs- und Endpunkte der Wertschöpfungskette liegen. Bei der Bestimmung der Reihenfolge werden Levels für die einzelnen Stufen vergeben. Die höchste Stufe erhält hierbei Level 999, die niedrigste (Anfangspunkt) Level 1, bzw. 999 – Anzahl der Levels in der Kette. Wurden die Endpunkte der

[15] Quelle: interner Vortrag.
[16] Vgl. Turau, Volker; Friedrich, Vogel (2004), S. 25f.

Wertschöpfungskette erfolgreich identifiziert, werden diese vom Graphen gestrichen und erneut Anfangs- und Endpunkt bestimmt. In diesem Schritt erhält die Endstufe Level 998 und die Anfangsstufe Level 2. Dieser Algorithmus wiederholt sich, bis die Wertschöpfungskette komplett durchlaufen und jedem Knoten eine Stufe zugeordnet wurde. Während der Sequence können Rekursionen (s. 2.1) auftreten. Diese werden von der Sequence erkannt und die Kreisbeziehung an der Kante mit der kleinsten Menge aufgeschnitten. Um die Rekursion an der Wareneingangsseite bewerten zu können, wird der Werteansatz der Vorperiode verwendet und auf die aktuelle Menge proportionalisiert.

Das Costing, welches im nächsten Kapitel beschrieben wird, weiß nun, in welcher Reihenfolge es die einzelnen Knoten rechnen muss. Das Costing arbeitet hierbei Bottom-Up von den Rohstoffen zum Endprodukt. Der richtige Sequence-Level stellt also die korrekte Wertewälzung der Einsatzmaterialien zum Endprodukt sicher. Die folgende Abbildung (s. Abbildung 7) soll hierbei die Vorgehensweise der Sequence verdeutlichen.

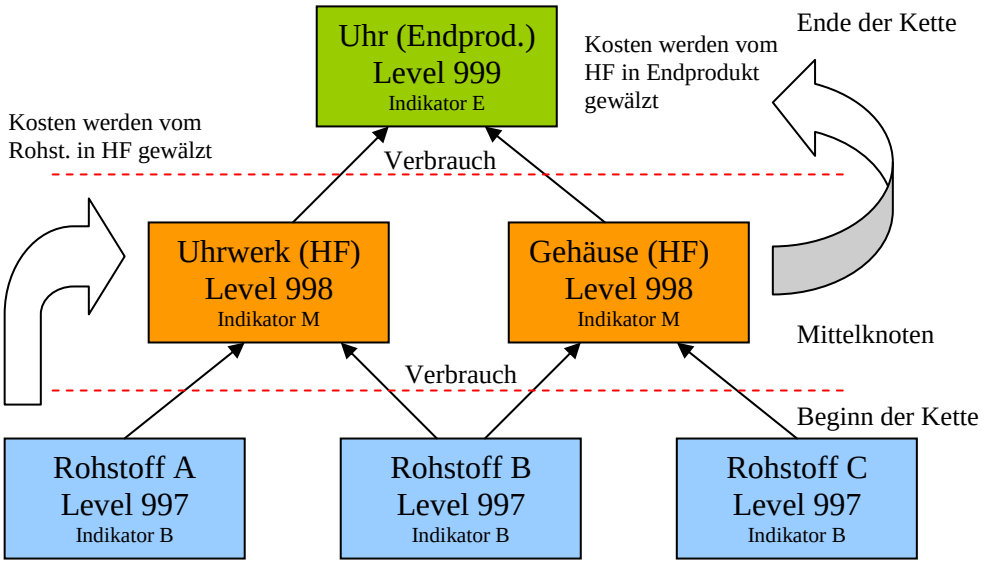


Abbildung 7 Wertschöpfungsgraph nach Sequence

Die Abbildung zeigt eine exemplarische Wertschöpfungskette, die stark vereinfacht, aus drei Levels besteht. Die Sequence bestimmt durch das Prüfen weiterer Vorgänger bzw. Nachfolger die Endpunkte der Wertschöpfungskette. Diese Endpunkte werden mit einer speziellen Markierung versehen (Indikator E, B). Stehen die Endpunkte fest, schneidet der Algorithmus diese Knoten ab und bestimmt erneut die Anfangs- und Endpunkte. Diese erhalten nun aber den Indikator M für Mittelknoten, da ausgeblendete Vorgänger oder Nachfolger vorliegen. Als Level bekommt die Mittelstufe 998, da diese den unmittelbaren Vorgänger der letzten Stufe 999 darstellt. Die Bestimmung der Reihenfolge ist wichtig, da so das Costing die Herstellkosten in der richtigen Reihenfolge von unten nach oben bestimmen kann. Bevor also das Halbfabrikat Uhrwerk kalkuliert werden kann, müssen zunächst die Rohstoffe gerechnet werden, die in dieses Halbfabrikat einfließen. Die einzelnen Costing Items werden also aufsteigend nach dem Level sortiert und danach vom Costing gerechnet (s. 2.3.4). Die einzelnen Materialien in den Werken umfassen nun als weitere Information den Level (s. rechts in den Knoten der Abbildung 8).

2.3.4 Costing

Im letzten Schritt, dem Costing, wird die Wertschöpfungskette in der durch die Sequence bestimmten Reihenfolge abgearbeitet, um die Herstellkosten unter Einbezug der Informationen aus den Financial Items zu ermitteln. Als Ausgangspunkt hierfür dienen Endbestände der jeweiligen Materialien aus der Vorperiode. Diese werden als Anfangsbestand der zu kalkulierenden Periode gesetzt und anschließend um die Zugänge erweitert. Der daraus ermittelte kumulierte Bestand dient nachfolgend der Bewertung der Abgänge und schließlich der Bestimmung des neuen Endbestands. Dieses Schema wird soll in der Abbildung 8 verdeutlicht werden.

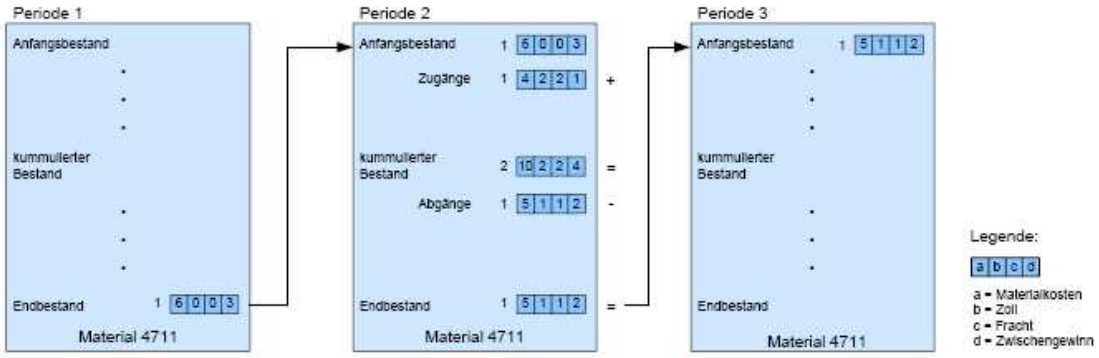


Abbildung 8 Vorgehensweise Costing[17]

Führt man diese Endbestandsermittlung für alle Materialien entlang der Wertschöpfungskette durch, so lassen sich alle für ein Endprodukt aufgetretene Herstellkosten, Frachtkosten und Zwischengewinne ausweisen. Die folgende Abbildung (s. Abbildung 9) zeigt den Wertschöpfungsgraph nach dem Costing mit zugewiesenen Werten.

[17] Quelle interner Vortrag.

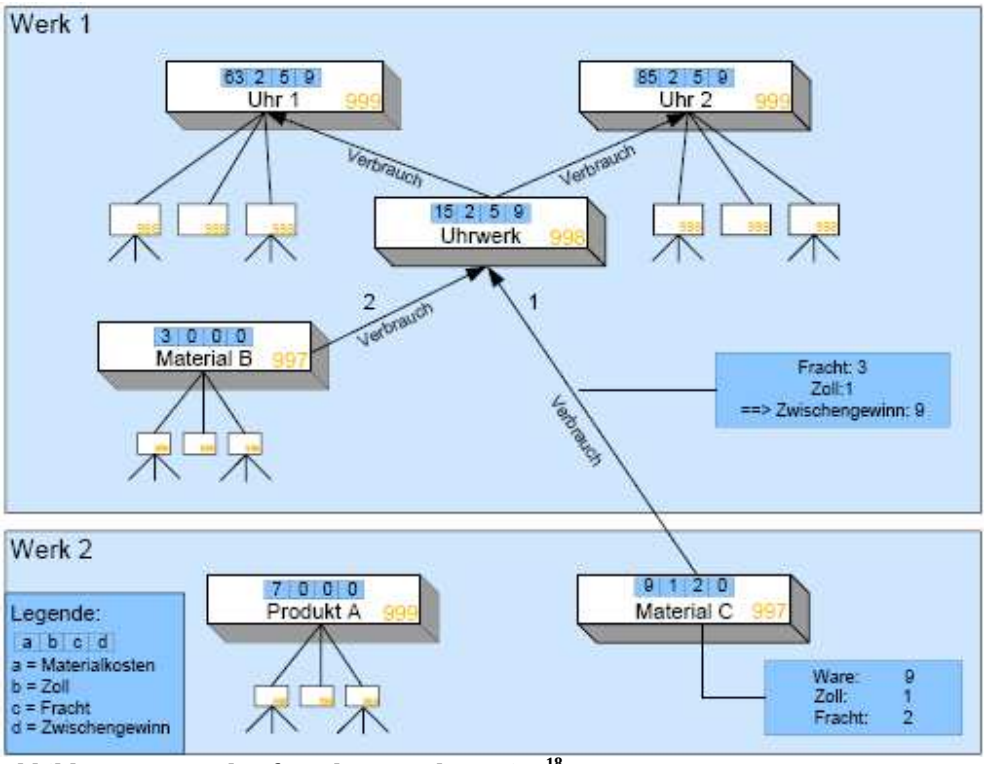


Abbildung 9 Wertschöpfungskette nach Costing[18]

Abbildung 9 zeigt die Auswirkungen des Costing auf den Graphen. Die Kostenelemente der einzelnen Costing Items sind nun mit Werten gefüllt. Ab dem werksübergreifenden Verkauf entsteht der Zwischengewinn durch die Differenz des Verkaufspreises von Werk 1 und den tatsächlichen Herstellkosten. Der Erlös von Werk 1 an Werk 2 für den Verkauf von Material C beträgt 25 GE, wobei in diesem Erlös Zölle und Frachten enthalten sind, die in die Fracht und Zoll Kostenelemente des Uhrwerks einfließen. Da die Kosten von Werk 1 für Material C inklusive bisheriger Frachten und Zölle 12 betragen, entsteht ein Zwischengewinn von 9 GE. Innerhalb eines Werkes werden so alle Kostenelemente eines Materials, bei einem Verbrauch, in das empfangende Material gewälzt.

Nach dem Costing hat die GCP alle notwendigen Werte ermittelt, um eine Analyse der globalen Kosten durchzuführen. Hierbei kann nun eine Aussage über die tatsächlich angefallenen Herstellkosten getroffen werden. Die globalen Herstellkosten sind um den Zwischengewinn, der innerhalb des Konzerns angefallen ist, bereinigt. Operative und strategische Entscheidungen basieren nun auf einem realistischeren Werteansatz, der das Konzernergebnis durch mehr Transparenz verbessern kann.

In der Grafik (s. Abbildung 9) ist ein Ausschnitt aus der globalen Wertschöpfungskette, komplettiert mit den Werten des Costings, dargestellt. Beim Nachvollziehen der Vorgehensweise des Costing ist ersichtlich, dass eine solche Grafik für das Verständnis des Beispiels eine sehr wichtige Rolle spielt. Es ist also nicht nur sinnvoll sondern auch erforderlich, den Anwender durch eine solche Grafik bei der Analyse der Ergebnisse zu unterstützen. Die vorliegende Arbeit setzt an dieser Stelle an und beschreibt die

[18] Quelle interner Vortrag.

Implementierung einer Anzeige von globalen Wertschöpfungsketten in einem SAP System.

2.4 XML

Die Extensible Markup Language (XML) ist eine Markup Sprache zur strukturierten Darstellung von Daten in Textdokumenten. Der größte Vorteil ist hier, dass diese Sprache sowohl von Mensch als auch Maschine lesbar ist.
Hierbei beschreibt XML auch einen Dokumentverarbeitungsstandard des W3C (World Wide Web Consortium), das auch Standards wie z.B. HTML (Hypertext Markup Language) verwaltet und herausgibt. Die Basis von XML bildet die Sprache SGML (Standard Generalized Markup Language), die jedoch weitaus komplexer ist als XML. Den Durchbruch schaffte XML als Jon Bosak, Mitarbeiter von Sun Microsystems, eine W3C Arbeitsgruppe bildete, die aus SGML eine Sprache ableiten sollte, die in ihrer Form für das Internet geeignet ist. Im Gegensatz zu HTML erlaubt XML die Erstellung eigener Markups, deren Grammatik in Dokumenttypdefinitionen und Stylesheets näher beschrieben werden kann.[19]

Im folgenden Abschnitt sollen nun einige Techniken kurz vorgestellt werden, die im weiteren Verlauf der Arbeit verwendet werden.

Ein XML-Schema definiert die Grammatik und Struktur eines XML-Dokuments. Ein Dokument ist nur dann gültig, wenn es alle Vorgaben des Schemas erfüllt. Diese Validierung stellt somit sicher, dass ein gültiges Dokument immer den erwarteten Inhalt liefert. Es folgt ein einfaches Beispiel:

Tabelle 1 Beispiel XML-Dokument mit Schema

XML-Dokument	`<?xml version="1.0" encoding="iso-8859-1"?>` `<person gender="M">Hans</person>`

XML-Schema	`<xs:complexType>`
	`<xs:element name="person">`
	`<xs:attribute name="gender" type="gender"/>`
	`<xs:simpleType name="gender">`
	`<xs:restriction base="xs:string">`
	`<xs:pattern value="M|F"/>`
	`</xs:restriction>`
	`</xs:simpleType>`
	`</xs:comlextype>`

Dieses Beispiel zeigt ein gültiges XML-Dokument mit dem zugehörigen Schemaauszug. Es wird festgelegt, dass im Attribut „gender“ des Elements „person“ nur ein „M“ oder „F“ stehen darf. Neben der Validierung dienen XML-Schemas der Dokumentation, der Unterstützung von Abfragen, der Datenbindung und dem geführten Editieren von XML-Dokumenten.[20]

Man kann ebenso wohlgeformte, d.h. dem Standard entsprechende XML-Dokumente schreiben, ohne eine Grammatik zu definieren. Hiezu muss das Dokument jedoch wohlgeformt (engl. well-formed) sein. Ein Dokument ist dann wohlgeformt, wenn es hierarchisch ist. Diese Hierarchie lässt sich übersichtlich in einer Baumstruktur

[19] Vgl. Eckstein, Robert; Casabianca Michel (2002), S. 5.
[20] Vgl. Van der Vlist, Eric (2002), S. 2ff.

darstellen. Diese Darstellung nennt sich DOM (Document Object Model). Um von eigenen Programmen auf ein XML-Dokument zugreifen zu können, benötigt man eine entsprechende Programmierschnittstelle. Einige Programmiersprachen bieten hier Schnittstellen an, um Modifikationen des DOM zu ermöglichen. Auch HTML-Dokumente werden in einem DOM abgebildet, der bei Dynamic HTML von JavaScript modifiziert werden kann.

Auch proprietäre Programmiersprachen wie ABAP stellen an dieser Stelle bestimmte Bibliotheken zur Verfügung, um einen solchen Dokumentbaum zu erstellen bzw. zu editieren. Im Falle dieses Projekts wurde die SAP IXML Bibliothek verwendet, um ein XML-Dokument neu aufzubauen. Hierbei stellt die SAP im Klassenmodell bestimmte Standardmethoden zur Verfügung, um dem Dokument Elemente und Attribute hinzuzufügen. Neben diesen Methoden zur Bearbeitung des eigentlichen Dokuments, werden auch Funktionen angeboten, die das Dokument in eine externe Datei umwandeln oder über Parser externe Dateien in einen DOM Baum einlesen können.

2.5 HTML

HTML ist wie XML eine Auszeichnungssprache jedoch mit einer festen Vorschrift der zugelassenen Markups. Die heute gängigen Browser stellen ein gewisses Set an Markups zur Verfügung, die interpretiert werden können. Die aktuell gültige HTML-Spezifikation des W3C ist 4.01, wobei das Konsortium vor kurzem einen Entwurf zur neuen Version 5.0 veröffentlicht hat.[21]

HTML basiert wie XML auf der Markup Sprache SGML, wobei zugunsten der Benutzerfreundlichkeit auf einige Funktionen von SGML verzichtet wurde. Mit HTML können bestimmte Regeln und Formate erstellt werden, wie Inhalte auf dem Bildschirm bzw. in einem Browser angezeigt werden sollen. Diese Regeln stehen in Form einer Syntax mit bestimmten Markup-Tags zur Verfügung. Hierbei können die Dokumente mit Hilfe von Hyperlinks auch interaktiv gestaltet und verlinkt werden.[22]

Im konkreten Fall wurden in diesem Projekt HTML-Seiten als Container für Java Applets vorgesehen. Die erstellte Container-HTML-Seite enthält einige Besonderheiten, wie z.B. versteckte Formulare zur Weiterleitung der Events aus dem Java-Applet und einige Javascript Methoden. Diese versteckte Form ist erforderlich, da innerhalb des SAP GUI ein HTML-Control als Container für diese Webseite verwendet wird. Die Form leitet die Eigenschaften des Events über das spezielle Formular an den SAP GUI weiter. Das HTML-Control ist ein Standard Steuerelement der SAP mit Methoden zur Navigation und Dokumentverarbeitung. Formulardaten werden in HTML über GET und POST-Methoden an den „Server“ weitergeleitet, wobei die Daten bei der GET-Methode einfach an die URL angehängt werden. Das genutzte versteckte Formular verwendet hingegen die Methode POST, bei der die Daten in einer gekapselten Struktur weitergegeben werden.

Die verwendeten JavaScript-Methoden beziehen sich auf die aus dem Applet erzeugten Events. JavaScript ist eine standardisierte Programmiersprache, deren Code innerhalb von Browsern bzw. HTML-Seiten abläuft. Die Syntax ist an Java angelehnt. JavaScript

[21] Vgl. http://www.w3.org/TR/html401/ eingesehen am 14.11.2007.
[22] Vgl. Musciano, Chuck; Kennedy, Bill (2003), S. 9.

erlaubt es, Webseiten dynamisch zu gestalten, erhöht aber auch das Sicherheitsrisiko und muss von den einzelnen Browsern unterstützt werden. Oftmals kommt es hierbei zu Kompatibilitätsproblemen zwischen verschiedenen Browsern. Javascript verfolgt einen objektorientierten Ansatz. Elemente einer HTML Seite oder die Seite selbst werden über DOM Elemente angesprochen.[23] Die häufigste Anwendung findet JavaScript bei clientseitigen Prüfungen von Formulardaten oder der aufblühenden AJAX-Technologie.

2.6 Java

Java ist eine eigenständige objektorientierte Programmiersprache und wird in der aktuellen Version 6.0 von der Firma Sun Microsystems angeboten und weiterentwickelt. Da JNet, die im Projekt verwendete Anzeigetechnologie, komplett in Java implementiert ist, soll in diesem Abschnitt eine kurze Vorstellung erfolgen.

Die Besonderheit von Java ist die Plattformunabhängigkeit, da die Programme zunächst in Bytecode für die virtuelle Maschine übersetzt werden. Dieser Code wird dann auf der virtuellen Maschine ausgeführt und für die jeweilige Plattform angepasst. Anwendungen laufen somit sowohl in einer Windows als auch Linux Umgebung und sehen dabei fast identisch aus.[24] Einen weiteren Vorteil von Java bilden die Kosten, da viele Entwicklungsumgebungen sowie die Entwicklungspakete kostenlos verfügbar sind. Weiterhin gibt es eine Fülle von Werkzeugen zur Bearbeitung von XML-Daten mit Java.

Java gibt es in mehreren Varianten. Für einfachere Programme gibt es die Java Standard Edition. Will man jedoch komplexe Business-Applikationen entwickeln, benötigt man die Java Enterprise Edition. Die jeweiligen Pakete liefern die API und die entsprechenden Compiler. Die beliebtesten Entwicklungsumgebungen für Java sind Eclipse und NetBeans von Sun. Um Java Programme ausführen zu können, braucht ein Benutzer nicht das komplette Entwicklungspaket zu installieren, sondern nur die erforderliche Laufzeitumgebung JRE.

Da Java objektorientiert ist, werden auch fast alle Paradigmen der Objektorientierung angeboten. Diese sind bspw. Vererbung, Polymorphie, Klassen, Interfaces und Packages. Seit dem Basis Release 4.7 kann auch in SAP-Umgebungen mit Java Programmiert werden, da der SAP WebAS die entsprechenden Schnittstellen bereitstellt.

Im vorherigen Abschnitt war von Java-Applets innerhalb einer HTML-Seite die Rede. Ein Applet ist eine Java-Applikation, die innerhalb eines Browsers abläuft. Die Programmstruktur normaler ausführbarer Java-Programme und Applets unterscheidet sich leicht. Damit ein Applet in einem Browser laufen kann, muss dieser eine Virtuelle Maschine für Java installiert haben. Die VM stellt eine abgeschottete Laufzeitumgebung dar, die das Sicherheitsrisiko minimieren soll und auch „Sandbox“ genannt wird. Ein Applet kann über ein spezielles HTML-Tag in die Seite eingebunden werden

[23] Vgl. Flanagan, David (2006), S. 1.
[24] Vgl. Ullenbom, Christian (2007), S. 2. (http://www.galileocomputing.de/openbook/javainsel7/javainsel_01_003.htm#mj549b65f06b9894115fd8b04f8cb898f5)

3 Spezifikation und Anforderungen

Die Entwicklung eines Werkzeuges zum Anzeigen globaler Wertschöpfungsketten bringt viele Anforderungen aus den verschiedensten Bereichen mit sich. Diese können sowohl betriebswirtschaftlich als auch technisch definiert sein.

Ziel dieses Kapitels ist es dem Leser zunächst die primären Use-Cases einer Visualisierung globaler Wertschöpfungsketten vorzustellen. Aus den primären Use-Cases leiten sich viele unterstützende Use-Cases ab, die im weiteren Verlauf des Abschnitts in funktionale und nicht funktionale Anforderungen überführt werden. Hierbei sollen Grafiken dabei helfen, die Anforderungen in verschiedene Kategorien zu unterteilen.

3.1 Allgemeine Anforderungen

Die Anforderungen an eine Softwarekomponente lassen sich in funktionale und nicht-funktionale Aspekte unterteilen. Das Untersuchen der Anforderungen bei Software Projekten bezeichnet man auch als Requirements Engineering. Im folgenden Abschnitt werden die allgemeinen Anforderungen vorgestellt. Die primären Funktionen sollen zunächst in einem Use-Case-Diagramm dargestellt werden, um die direkten Funktionen zu beschreiben, die die Visualisierung globaler Wertschöpfungsketten zur Verfügung stellen soll:

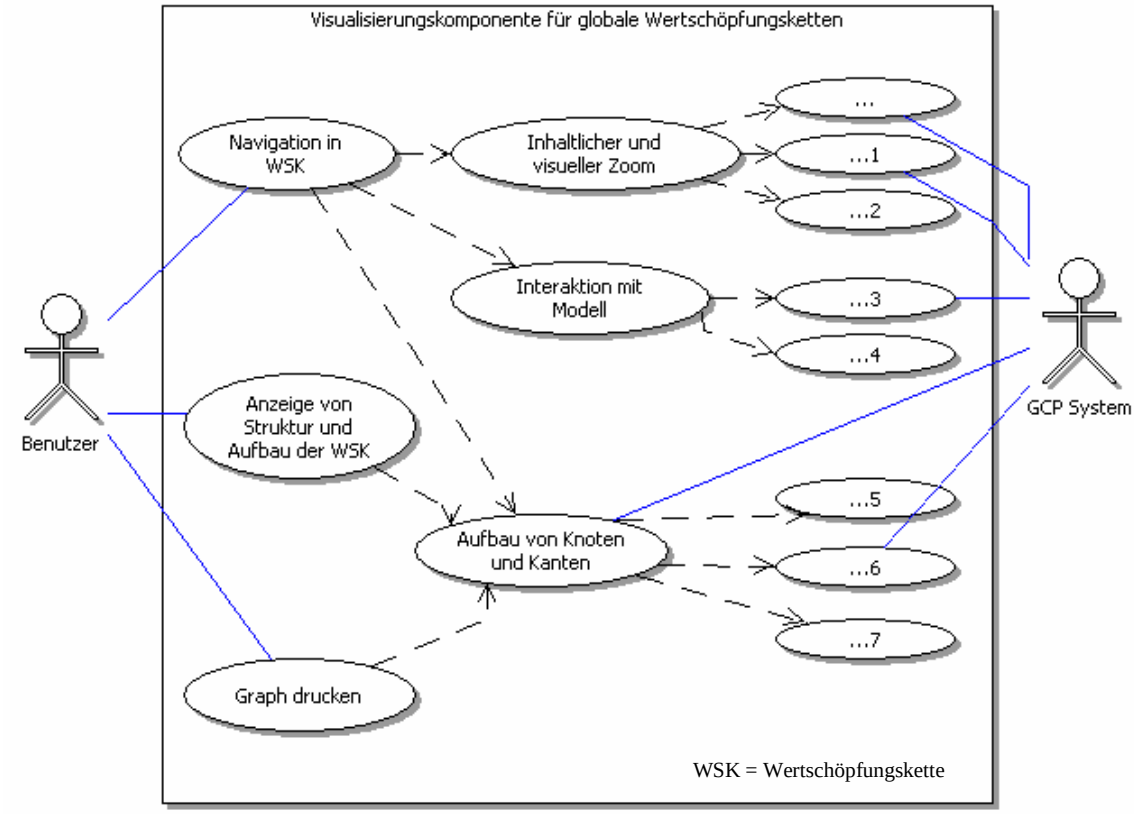


Abbildung 10 Use-Case-Diagramm „Visualisierung globaler Wertschöpfungsketten“.

Die Abbildung zeigt ein Use-Case-Diagramm mit den primären und sekundären Use-Cases. Auf der rechten und linken Seite der Grafik sind die beteiligten Aktoren dargestellt, die zum einen aus dem Benutzer, und zum anderen aus dem GCP System

bestehen. Die primären Use-Cases bestehen aus der Anzeige der Wertschöpfungsstruktur, also des eigentlichen Graphen, dem Drucken dieses Graphs und der Navigation durch die Wertschöpfungskette. Die drei primären Use-Cases benötigen einige weitere Funktionen, die in der Mitte der Grafik dargestellt sind. Diese sog. Sekundären Use-Cases bilden den Überbegriff weiterer einzelner Funktionalitäten, die, links in der Grafik, exemplarisch dargestellt wurden. Um die einzelnen Funktionalitäten übersichtlich darstellen zu können, wurde eine alternative Darstellungsform der Anforderungen gewählt. Hierbei werden die Funktionen in einzelne Funktionsgruppen unterteilt, die den sekundären Use-Cases entsprechen. Diese Funktionsgruppen beinhalten die jeweiligen Anforderungen im Detail. Im weiteren Verlauf dieses Kapitels werden die Anforderungen anhand einer solchen Hierarchiegrafik (s. Abbildung 11) vorgestellt und beschrieben.

3.1.1 Funktionale Anforderungen

Die funktionalen Anforderungen bilden das Fundament der Software Spezifikation. Sie bilden den wichtigsten Bestandteil einer Software, da diese zunächst an der Erfüllung der funktionalen Anforderungen gemessen wird. Die folgende Grafik (s. Abbildung 11) soll die allgemeinen funktionalen Anforderungen, geordnet nach Kategorie, darstellen. Diese Darstellung soll helfen, einen Überblick über die Vielzahl von sekundären Use-Cases zu erhalten und diese in Funktionsgruppen einzuteilen.

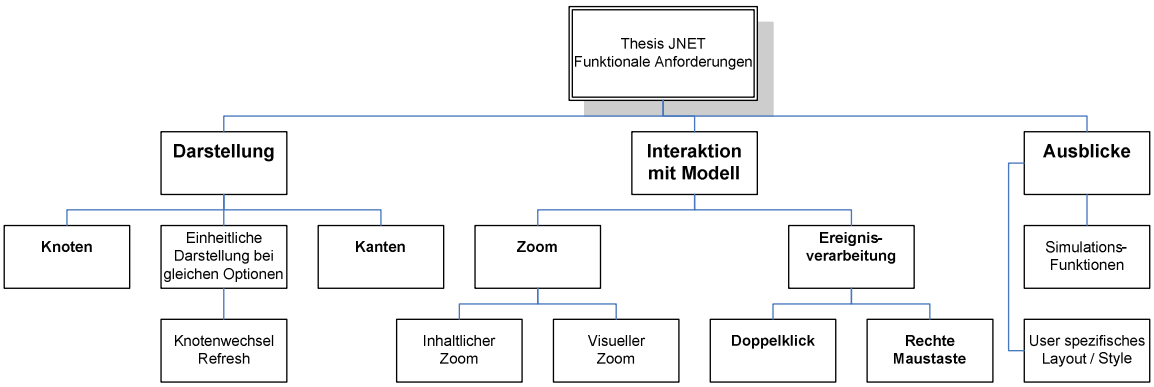


Abbildung 11 Allgemeine funktionale Anforderungen

Aus den primären Use-Cases (s. linke Seite der Abbildung 10) lassen sich die drei Hauptfunktionsgruppen Darstellung, Interaktion und Ausblick ableiten, die ihrerseits aus vielen weiteren Funktionalitätsanforderungen bestehen. Im folgenden Abschnitt sollen zunächst die primären und sekundären Use-Cases beschrieben werden.

Die erste Funktionsgruppe beinhaltet den Aufbau, bzw. die Darstellung eines für eine Wertschöpfungskette geeigneten Graphen. Die Software muss in der Lage sein, die Wertschöpfungskette als Netzgraphen darzustellen. Dabei sollen Knoten- und Kantenbeziehungen in Hierarchien visualisiert werden. Sollten alle Knoten eine gemeinsame Eigenschaft besitzen, wird diese in einem gesonderten Bereich des Bildschirms dargestellt. Weiterhin muss die Software bei einem Wechsel des Knotens durch ein internes oder externes Ereignis einen neuen Graphen für den gewählten Knoten, der einen neuen Ausgangspunkt darstellt, erstellen. Es erfolgt ein Wechsel des Kontexts.

Die zweite Funktionsgruppe umfasst die Interaktion mit dem Benutzer. Im Graphen sollen dem Benutzer verschiedene Funktionen zur Verfügung stehen, um im Modell zu navigieren, oder bestimmte Zusatzinformationen anzuzeigen. Die Interaktion unterteilt sich dabei in zwei weitere Kategorien. Dies sind zum einen die Zoomfunktionen, die sich nochmals in einen inhaltlichen und einen visuellen Zoom unterscheiden, und zum anderen die Ereignisse, die durch einen Doppelklick bzw. einen Klick auf die rechte Maustaste ausgelöst werden. Die Zoomfunktion soll dem Benutzer ermöglichen, weitere Knoten zum aktuellen Kontext hinzu zu laden oder einen bestimmten Level zu entfernen. Eine Unterscheidung der Detailstufe ist ebenso Teil der Zoomfunktion. Die Ereignisse dienen primär der Navigation im Graphen. Wird ein Knoten ausgewählt sollen Details zum Knoten angezeigt werden. Diese Detailanzeige ist bereits im Rahmen der GCP implementiert und nennt sich Costing Item Anzeige (/IMCGCP/50). Bei einem Klick auf die Kanten ist vorstellbar, in die Details der Materialbewegung zu navigieren und diese in einem Standard Anzeigewerkzeug z.B. ALV Browser anzuzeigen. An dieser Stelle wird für weitere Details auf das entsprechende Detail Kapitel verwiesen.

Die dritte Funktionsgruppe wird unter dem Punkt Ausblick zusammengefasst, da diese Anforderungen nicht Bestandteil des angefertigten Prototyps sind. Es sollen jedoch Schnittstellen und Erweiterungspunkte vorgesehen werden, damit diese Ideen in weiteren Projekten später ergänzt werden können. Die Visualisierung der Wertschöpfungskette soll individuell, pro Benutzer, durch Customizing angepasst werden können. Den wichtigsten Punkt bildet hier jedoch die grafische Steuerung von Simulationen. Autorisierte Benutzer können mit dieser Funktionalität durch Änderungen am Graph und der Wertschöpfungsstruktur direkt die Daten der GCP modifizieren und somit bestimmte Zustände testen. Weiteres im Abschnitt Ausblick 7.3 bzw. dem Detailkapitel 3.5.1.

Die folgende Tabelle (s. Tabelle 2) stellt eine Priorisierung der in der Abbildung dargestellten Anforderungen dar. Alle Funktionen, die in den Prototypen einfließen sollen, wurden mit einer Aufwandsschätzung versehen.

Tabelle 2 Priorisierung funktionale Anforderungen

Priorität	Anforderung	Aufwand (in MT)
1	Darstellung von Knoten und Kanten	12
2	Reaktion auf Klick/Doppelklick	5
3	Inhaltlicher Zoom +/- von Level	2
4	Refresh bei Knotenwechsel	2
5	Visueller Zoom / Ändern der Detailstufe	6
6	Eigenes Kontextmenü bei Rechtsklick	4
7	Einheitliche Darstellung gleicher Eigensch.	1
8	Eigenes Layout Customizing	3
9	Simulationsfunktion	Ausblick

3.1.2 Nicht-funktionale Anforderungen

Die nicht funktionalen Anforderungen eines Systems bilden einen zweiten Punkt der Anforderungserhebung. Sie beschreiben nicht direkt, welche Funktion eine Software durchführen muss, sondern definiert, wie die Funktionen arbeiten. Die nicht-funktionalen Anforderungen sind allgemeiner definiert als die einzelnen echten Funktionen der Software. Ein Beispiel bildet hier die Aussage über die Qualität und Performance eines Produkts mit der resultierenden Fragestellung: In welcher Zeit wird eine Funktion der Software abgearbeitet? Die nicht-funktionalen Anforderungen beziehen sich dabei nicht auf die primären und sekundären Use-Cases, da sie eben keine direkte Auswirkung auf die Funktion der Software haben und dem Anwender zunächst verborgen bleiben.

Die folgende Abbildung zeigt die nicht-funktionalen Anforderungen an die IT- gestützte Visualisierung einer globalen Wertschöpfungskette:

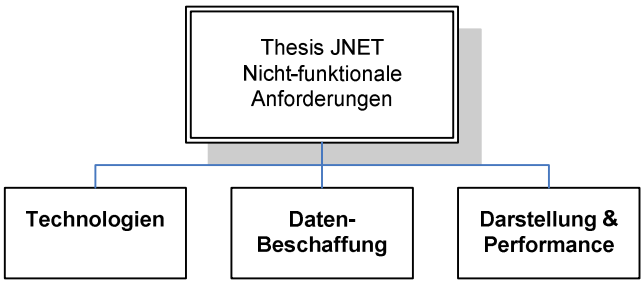


Abbildung 12 nicht-funktionale Anforderungen

Für die vorliegende Themenstellung wurden drei weitere Hauptgruppen von nicht-funktionalen Anforderungen definiert. Die erste Gruppe bilden die Technologien. Hier werden alle nicht-funktionalen Anforderungen zusammengefasst, die die technische Unterstützung betreffen. Die zweite Gruppe bildet die Datenbeschaffung. Dieser Punkt beinhaltet Vorgaben an die Datenauswahl aus dem Quellsystem. Der letzte Punkt behandelt die Darstellung und Performance der Visualisierungssoftware. Hier werden Vorgaben über das Aussehen der Knoten und Kanten getroffen. Die Performance ist für Echtzeit Programme wie eine GUI ebenfalls sehr wichtig und sollte zufrieden stellende Ergebnisse liefern.

Neben den genannten nicht-funktionalen Anforderungen gelten die allgemeinen Anforderungen an eine Software wie z.B. Wartbarkeit durch Kommentare und lesbares Coding, Qualität und Skalierbarkeit.

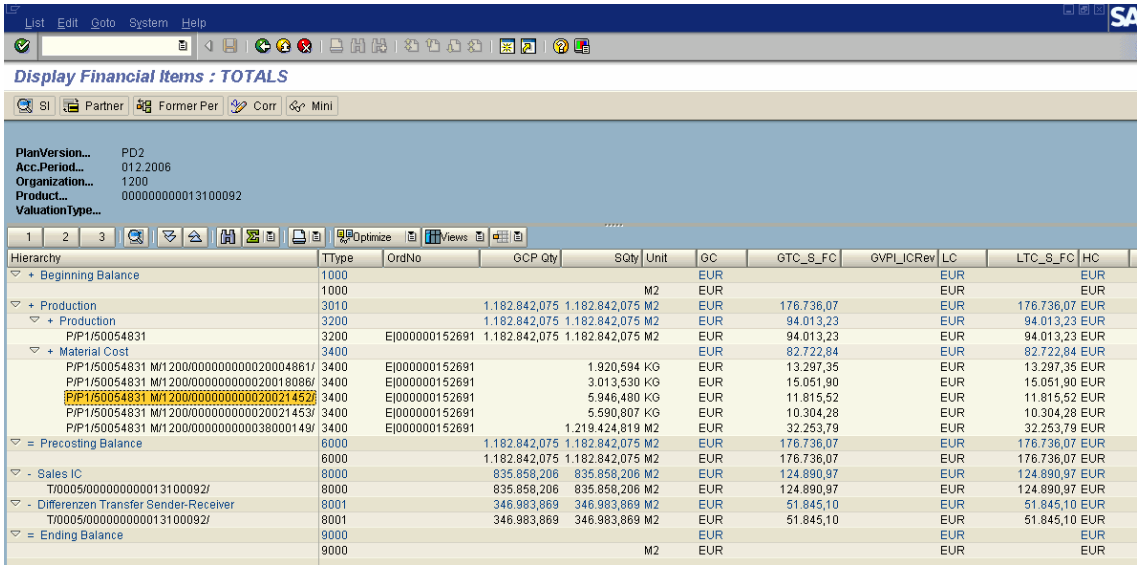
Auch an dieser Stelle wird für weitere Informationen auf das Detailkapitel 3.5.2 verwiesen. Dort werden die einzelnen Anforderungsgruppen aufgegriffen und näher erläutert.

3.2 Integration in die Anwendungsumgebung

Die Komponente zur Navigation bzw. zur Anzeige globaler Wertschöpfungsketten soll nicht alleine lauffähig sein, sondern mit den GCP Komponenten verknüpft werden bzw. mit diesen kommunizieren. Als ein exemplarisches Beispiel soll hier die GCP Anzeigefunktion für CIs genannt werden (s. Abbildung 13). Diese Funktion stellt alle

zusammengehörigen FI in einem ALV Tree Control dar. In dieser GCP Applikation ist eine Navigation durch die Wertschöpfungskette sehr sinnvoll, da der Anwender somit schnell alle verbundenen Partner identifizieren und auswählen kann. Bisher können die Partner CI über einen Doppelklick auf eine Zeile in der Anzeigefunktion erreicht werden. Diese Navigation hat den Nachteil, dass immer nur über eine Stufe navigiert werden kann.

Die Abbildung zeigt die erste Applikation, in die der Prototyp eingebunden werden soll, auf dem derzeitigen Stand. Durch einen Klick auf die markierte Zeile könnte man in das entsprechende Partnermaterial navigieren. Diese Applikation soll durch eine graphische Darstellung der Umgebung des Wertschöpfungsgraphen für ein Material erweitert werden. Die zu entwickelnde Komponente soll hier in der linken Seite eingefügt werden und eine Hierarchie über mehrere Stufen darstellen, um die Navigation zu verbessern. Bei der Auswahl eines Knoten in diesem Navigationsfenster soll die dem Knoten entsprechende Anzeigefunktion auf der rechten Seite des Bildschirms eingeblendet werden.



Hierarchy	TType	OrdNo	GCP Qty	SQty	Unit	GC	GTC_S_FC	GVPI_ICRev	LC	LTC_S_FC	HC
+ Beginning Balance	1000					EUR			EUR		EUR
	1000				M2	EUR			EUR		EUR
+ Production	3010		1.182.842,075	1.182.842,075	M2	EUR	176.736,07		EUR	176.736,07	EUR
+ Production	3200		1.182.842,075	1.182.842,075	M2	EUR	94.013,23		EUR	94.013,23	EUR
P/P1/50054831	3200	E|000000152691	1.182.842,075	1.182.842,075	M2	EUR	94.013,23		EUR	94.013,23	EUR
+ Material Cost	3400					EUR	82.722,84		EUR	82.722,84	EUR
P/P1/50054831 M/1200/00000000000200048617	3400	E|000000152691		1.920,594	KG	EUR	13.297,35		EUR	13.297,35	EUR
P/P1/50054831 M/1200/00000000000200180867	3400	E|000000152691		3.013,530	KG	EUR	15.051,90		EUR	15.051,90	EUR
P/P1/50054831 M/1200/00000000000200214527	3400	E|000000152691		5.946,480	KG	EUR	11.815,52		EUR	11.815,52	EUR
P/P1/50054831 M/1200/00000000000200214537	3400	E|000000152691		5.590,807	KG	EUR	10.304,28		EUR	10.304,28	EUR
P/P1/50054831 M/1200/00000000003800014907	3400	E|000000152691		1.219.424,819	M2	EUR	32.253,79		EUR	32.253,79	EUR
= Precosting Balance	6000		1.182.842,075	1.182.842,075	M2	EUR	176.736,07		EUR	176.736,07	EUR
	6000		1.182.842,075	1.182.842,075	M2	EUR	176.736,07		EUR	176.736,07	EUR
- Sales IC	8000		835.858,206	835.858,206	M2	EUR	124.890,97		EUR	124.890,97	EUR
T/0005/000000000013100092/	8000		835.858,206	835.858,206	M2	EUR	124.890,97		EUR	124.890,97	EUR
- Differenzen Transfer Sender-Receiver	8001		346.983,869	346.983,869	M2	EUR	51.845,10		EUR	51.845,10	EUR
T/0005/000000000013100092/	8001		346.983,869	346.983,869	M2	EUR	51.845,10		EUR	51.845,10	EUR
= Ending Balance	9000					EUR			EUR		EUR
	9000				M2	EUR			EUR		EUR

Abbildung 13 GCP Anzeigefunktion YG50

Das Navigationsfenster soll hierbei nicht von Beginn eingeblendet werden sondern über einen Button abrufbar sein. Nach diesem Prinzip sollen auch weitere Applikationen eingebunden werden können.

3.3 Benutzerrollen

Dieser Abschnitt soll die Bedeutung der Endbenutzer vorstellen. Der Endbenutzer tritt in der Rolle eines normalen SAP bzw. GCP Benutzers auf, der so schnell und einfach wie möglich an die Informationen bzw. Ergebnisse der GCP Kalkulation kommen möchte. Im weitesten Sinne sind es die Benutzer, die mit den Reporting Funktionen vertraut sind. Für technische Einzelheiten besteht hier kein Interesse. Anwenderfreundlichkeit, Korrektheit der Daten und Performance spielen für diese Anwendergruppe eine wesentlich größere Rolle.

Im konkreten Fall des Projekts hat der Endanwender genau eine Schnittstelle zu bedienen (s. Kapitel 3.6). Der Teil der Datenbeschaffung bzw. wie genau ein Event verarbeitet wird, bleibt dem Benutzer verborgen. Der Benutzer logt sich in die Anzeigefunktion ein und drückt den Button zum Öffnen der Navigationsfunktion. Danach kann er relevante Informationen aus dem Modell entnehmen oder per Doppelklick in die Anzeigefunktion des ausgewählten CI wechseln.

Sobald die Simulationsfunktion implementiert wurde, müssen für den Benutzer Berechtigungsprüfungen vorgenommen werden. Ab diesem Zeitpunkt ändert sich die Funktion des Benutzers. Er kann jetzt auch Modifikationen an Daten, die sensibel sein können, vornehmen. Das kann je nach dem Folgen für das ganze Modell nach sich ziehen. Ist ein Benutzer zur Simulation berechtigt, müssen bei diesem auch Kenntnisse über alle Funktionen vor dem Reporting vorhanden sein. Dies ist erforderlich, da diese Anwendungen nach einer Modifikation im Modell nochmals ganz oder zumindest teilweise laufen müssen.

3.4 Wiederverwendung von Software Komponenten

In GCP gibt es im Moment keine Anwendung, die eine Funktion ähnlich dieses Projekts erfüllt. Speziell für diesen Zweck gibt es also keine Komponenten, die direkt wieder verwendet werden können. Im Rahmen von ABAP Objects gibt es jedoch sehr viele Bibliotheken, die in vielen Programmen wieder verwendet werden können. In der Anzeigefunktion ist bereits eine Funktion zum Navigieren über eine Stufe eingebaut. Diese Funktion könnte sich auch die Navigationskomponente zu nutze machen, um in einen ausgewählten Knoten zu wechseln. Die Schnittstelle für den Absprung sollte also schon existieren. Es unterscheidet sich lediglich der Knoten, in den navigiert werden soll. Für einen generellen Einblick in alle wieder verwendbaren Komponenten der SAP, steht in einem R/3 System die Transaktion SE33(Reuse Library) zur Verfügung. In der SAP Reuse Library werden dem Entwickler bestimmte Standardobjekte, z.B. ein Button oder eine Tabellenelement, welche er für seine Zwecke genauer definieren kann. Hierbei kann das Objekt immer mit bestimmten Standardfunktionen modifiziert werden.

Viele der Komponenten und grafischen Elemente, die auch in diesem Projekt zum Einsatz kommen werden, finden sich in der SAP Reuse Library. Ein Beispiel sei hier der Docking Container als grafisches Element, der später das Navigationsmodell beinhalten soll. Für die Transformation von SAP Datentabellen in XML Daten stellt die SAP ebenso einige Bibliotheken zur Verfügung, die in mehreren Programmen zum Einsatz kommen können. Diese Komponenten haben den Vorteil, dass man sich selbst einige Programmierarbeit spart und somit das Rad nicht neu erfinden muss. Einen weiteren Vorteil bilden die detaillierten Dokumentationen zu den wieder verwendbaren Komponenten seitens der SAP. Für fast jede Bibliothek existiert eine Online Dokumentation mit einigen Programmierbeispielen.

3.5 Anforderungen im Detail

In diesem Abschnitt werden die im Kapitel 3.1 vorgestellten allgemeinen Anforderungen detailliert erläutert. Hierbei erfolgt ebenso eine Trennung der funktionalen und nicht-funktionalen Anforderungen.

3.5.1 Funktionale Anforderungen im Detail

Im Kapitel 3.1.1 wurden bereits die allgemeinen funktionalen Anforderungen vorgestellt. Diese wurden in verschiedene Funktionsgruppen unterteilt. Die einzelnen Anforderungen bzw. Features sind als sekundäre Use-Cases zu verstehen, die den primären Use-Case, Navigation in einer grafischen WSK, unterstützen bzw. diesen ermöglichen.

Anforderungen an die Darstellung einer globalen Wertschöpfungskette

Während der GCP-Läufe wird zur Kalkulation des Konzernergebnisses ein globaler Wertschöpfungsgraph erstellt, der jedoch nur in Form einer Datenstruktur vorliegt. Um eine solche Datenstruktur in ein graphisches Modell überführen zu können, wird eine passende Darstellungsform benötigt. Die unterschiedlichen Kalkulationsstufen sollen in einer Größer- / Kleinerbeziehung im Graphen angedeutet werden. Endprodukte und Halbfabrikate stehen also weiter oben in der Grafik als die Rohstoffe, die sich ganz unten befinden. Da es innerhalb der Kette zu Rekursionen kommen kann und die Waren in eine Richtung fließen, bedarf es eines gerichteten Graphen mit Zyklen. Hierbei liegt die Wertschöpfungskette nicht als Baum-, sondern als Netzstruktur vor. Innerhalb des Graphen sollen Knoten und Kanten angezeigt werden, wobei die Kanten eine Beziehung zwischen den Knoten abbilden. Die folgende Abbildung (s. Abbildung 14) zeigt den Anforderungsbereich „Darstellung“. In der untersten Stufe der Grafik finden sich die genauen Anforderungen an die Darstellung von Knoten und Kanten.

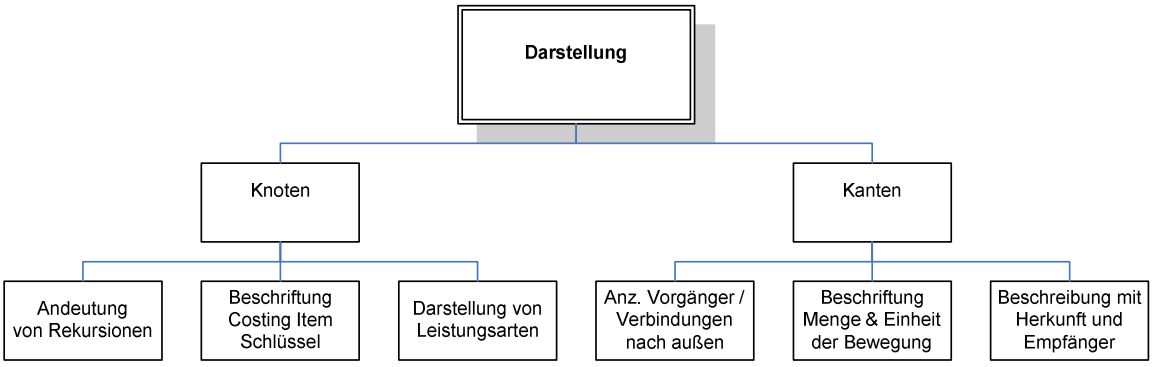
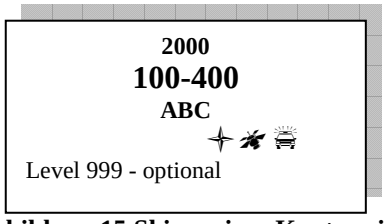
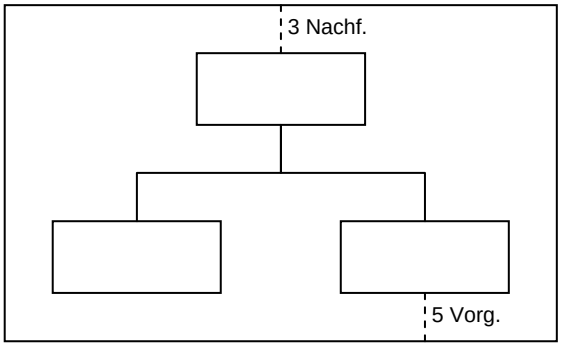


Abbildung 14 Funktionale Anforderungen "Darstellung"

Die folgende Tabelle beschreibt die Features, die die Software zur Verfügung stellen soll. Die einzelnen Features entsprechen den unteren Punkten der Abbildung 14.

Tabelle 3 Anforderung an die Darstellung von Wertschöpfungsketten

Feature	Beschreibung
Andeutung von Rekursionen	- Besondere Darstellung der Kanten für Rekursionen innerhalb des aktuellen Kontext - Markieren der Knoten, die eine Rekursion mit sich selbst enthalten - Feststellen der Rekursionen im kompletten Modell (WSK) und Andeutung der Rekursionen außerhalb des Kontexts (Partner liegt außerhalb der Anzeige)
Sprechende Beschriftung der Knoten mit	- Beschriftung mit Organisationseinheit, Produkteinheit und ggf. Bewertungsart (Produkteinheit hervorheben) - Anzeige von Informationen abhängig des aktuellen Zoomlevels

Feature	Beschreibung
dem Costing Item Schlüssel	(Produnit wird immer angezeigt) - Sequence Level kann optional eingeblendet werden - Berücksichtigen der Konvertierungsexits der Datenfelder - Icons für externe Lieferungen / Verkäufe vorsehen - Haben alle Knoten die gleiche Orgunit, soll die Orgunit in die Überschrift übernommen und in den Knoten ausgeblendet werden.  **Abbildung 15 Skizze eines Knotens im Modell** - Knoten soll entsprechend der Skizze aufgebaut sein. - Farbgestaltung in Anlehnung an Standard R/3
Darstellung von Leistungsarten	- Zusammensetzung der Wertschöpfung für einen Knoten betrachten (Anzeige der Wertschöpfung in einem sep. Graph) - Absprung in Leistungsartenanzeige in Kontextmenü vorsehen - Ermitteln einer geeigneten Darstellung für Leistungsarten (s. Skizze):  **Abbildung 16 Skizze zur Darstellung von Leistungsarten** - Rückwärtsnavigation in Wertschöpfungsgraph vorsehen
Anzeige weiterer Vorgänger und Nachfolger außerhalb des Kontexts	- Unterscheidung von End- und Mittelknoten - Bei Mittelknoten, die außerhalb des Kontext zeigen, sollen weitere Vorgänger und Nachfolger angedeutet werden - Ermitteln der Anzahl von Vorgängern und Nachfolgern - Die Skizze zeigt eine wie der Verlauf der WSK im Kontext angedeutet werden kann.  **Abbildung 17 Skizze Anzeige Vorgänger / Nachfolger**
Beschriftung der Kante mit	- Eine Kante beschreibt genau eine Beziehung zwischen zwei Knoten

Feature	Beschreibung
Menge und Mengeneinheit	- Darstellung des Volumens einer Kante (Gesamtvolumen für Periode) - Mengeneinheit für Volumen bestimmen und anzeigen - Evtl. Kantengewichtung vorsehen (Problematisch bezüglich unterschiedlicher Mengeneinheiten) - Alternative Darstellung für Kanten mit Nullmenge - Unterscheidung von Verbrauch, Transfer und Rekursion - Knotenvolumen ermöglicht keinen Rückschluss auf Losgröße (außer im Plan)
Beschreibung der Kante mit Sender und Empfänger	- Kante soll Kennzeichnung von Sender und Empfänger erhalten - Bessere Übersicht bei umfangreichem Modell - Kantenbeschreibung in Tooltip möglich oder eignes Kontextmenü - Sonderfall wenn eine Kante nach außerhalb zeigt

Anforderungen an die Zoom-Funktionen im Graph

Der Graph soll grundsätzlich zwei Arten des Zooms unterstützen. Es sollen, je nach Kontext, Daten nachgeladen werden können (inhaltlicher Zoom). Das visuelle Zoom nimmt Einfluss auf die Größe der im Graph angezeigten Elemente. Der kommende Abschnitt stellt die Zoomanforderungen genauer vor, die in der folgenden Abbildung übersichtlich dargestellt sind (s. Abbildung 18).

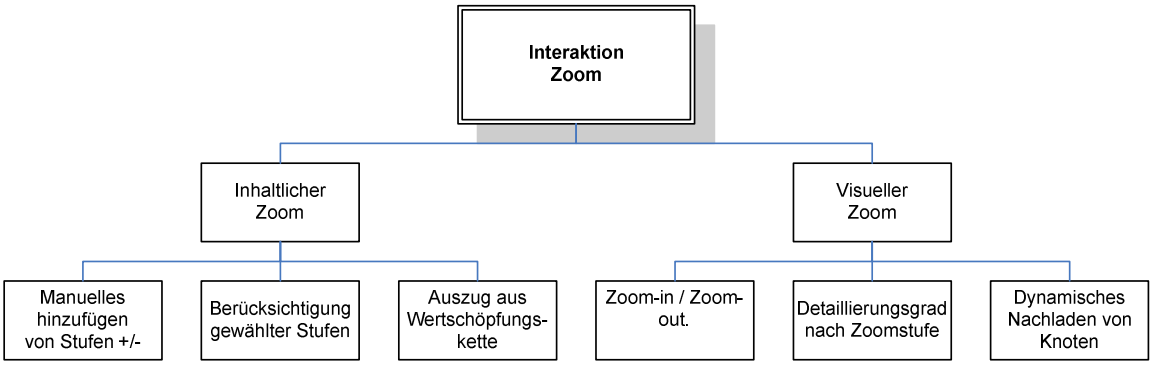


Abbildung 18 Funktionale Anforderungen "Zoom"

Die nun folgende Tabelle stellt die einzelnen Anforderungen der Funktionsgruppe „Interaktion/Zoom“ im Detail vor:

Tabelle 4 Anforderungen an das Zoom der Visualisierung

Feature	Beschreibung
Manuelles Hinzufügen von Stufen	- Stufen können während der Anzeige nachgeladen werden. - Stufen können in beide Richtungen ergänzt bzw. gelöscht werden - Kein Neuladen der ganzen Applikation erforderlich - Sonderbehandlung bei Rekursion (insbesondere beim Löschen)

Feature	Beschreibung
Berücksichtigung gewählter Stufen	- Standardwert für Stufen bereitstellen - Stufe gilt für beide Richtungen (TopDown – BottomUp) - Schema für Costing Item(0) (-2 – 1 0 + 1 + 2) bei 2 Stufen - Performante Selektion der Vorgänger und Nachfolger (Puffer) - Problematik bei BottomUp auf Rohstoffebene, falls diese in viele Halbfertigfabrikat einfließen - Knotenanzahl einschränken falls Anzeige nicht mehr sinnvoll
Auszug aus Wertschöpfungs-kette	- Anzeige der ganzen Kette oder eines Auszugs ermöglichen - Bei einem Auszug muss von einem Ausgangspunkt gestartet werden (CI als Parameter) - Umfang des Auszugs kann vom Benutzer bestimmt werden - Solange man sich in der gleichen Version bewegt, kein Neuladen der Applikation sondern Refresh (Wechsel des Ausgangspunkts)
Visuelles Zoom in / Zoom out	- Größenanpassung der Detailstufe - Standard Zoomfaktor setzen - Optimale Knotenanzahl für Bildschirmgröße - Minimalen bzw. maximalen Zoomlevel bestimmen - Übersichtsfenster für Kontext - Automatisches Scrollen auf Ausgangspunkt - Automatisches Nachladen von Knoten möglich, wenn Ende des scrollbaren Bereichs erreicht wurde
Variierender Detaillierungs-grad nach Zoom-faktor	- Zoomgewicht: Einzelne Elemente sollen, je nach aktuellem Zoomlevel, angezeigt oder ausgeblendet werden. - Jedem Element der Anzeige kann ein Zoomgewicht zugeordnet werden - Materialnummer bzw. Produkteinheit sollen immer angezeigt werden - Unterschiedliches Zoomgewicht für Kanten- und Knotenbeschriftung - Prüfen vorhandener Funktionalität
Dynamisches Nachladen von Knoten	- Puffertabelle für Version und Periode zum Beginn der Verarbeitung füllen - Relevanter Teil der Wertschöpfungskette + Puffer wird in Hauptspeicher abgelegt - Schnelleres Laden der Knoten und Kanten aus dem Hauptspeicher - Nachlademechanismus, falls Knoten oder Kante nicht im Puffer gefunden wurden - Puffertabelle soll mind. zwei Stufen mehr beinhalten als Stufen visualisiert werden sollen - Nähere Beschreibung des Mechanismus in der Tabelle folgenden Skizze (s. Abbildung 19)

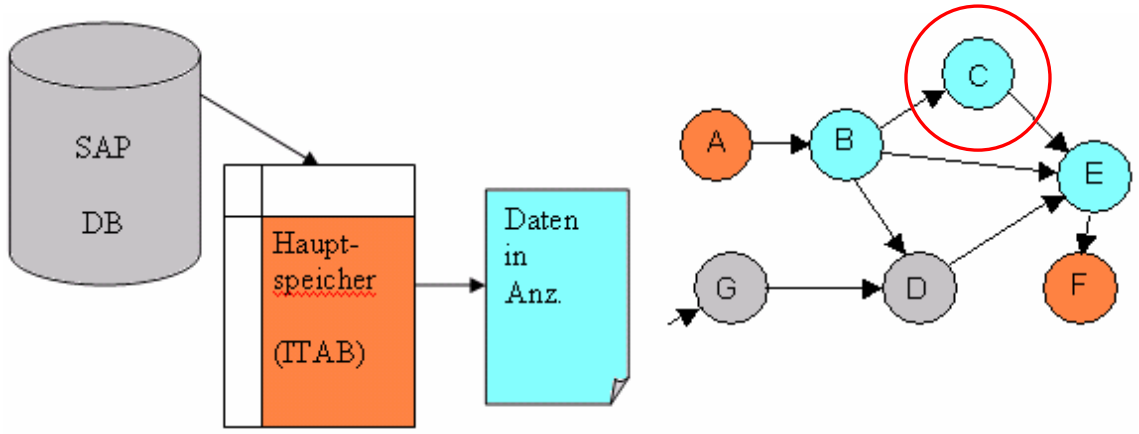


Abbildung 19 Nachladestrategie von Knoten

Die Abbildung zeigt rechts eine exemplarische Wertschöpfungskette mit den Knoten von A bis G. In diesem Falle sei der Bezugspunkt der Knoten C. Es sollen alle Knoten einen Level vor und einen Level nach diesem Knoten angezeigt werden (Knoten B, E). Die Selektion umfasst jedoch fünf Knoten (Bezugspunkt +- zwei Level) und legt diese in einer Puffertabelle (Hauptspeicher) ab. Dies hat den Vorteil, dass die Daten, bei einem Hinzufügen einer neuen Stufe, schneller zur Verfügung stehen. Wird also im aktuellen Kontext ein Level nach oben angefordert, wird der Knoten F aus dem Hauptspeicher geladen, und es ist keine weitere Datenbankselektion erforderlich.

Anforderungen an die Verarbeitung von Ereignissen

In der Anzeige von Wertschöpfungsketten in einem Graphen kann es zu mehreren Ereignissen kommen, insbesondere wenn eine Navigation ermöglicht werden soll. Die wichtigsten Ereignisse sind der Doppelklick und die Anforderung des Kontextmenüs mit der rechten Maustaste. Die folgende Abbildung zeigt die Funktionsgruppe der Ereignisverarbeitung. Hierbei sind die einzelnen Punkte in einer hierarchischen Übersicht dargestellt (s. Abbildung 20).

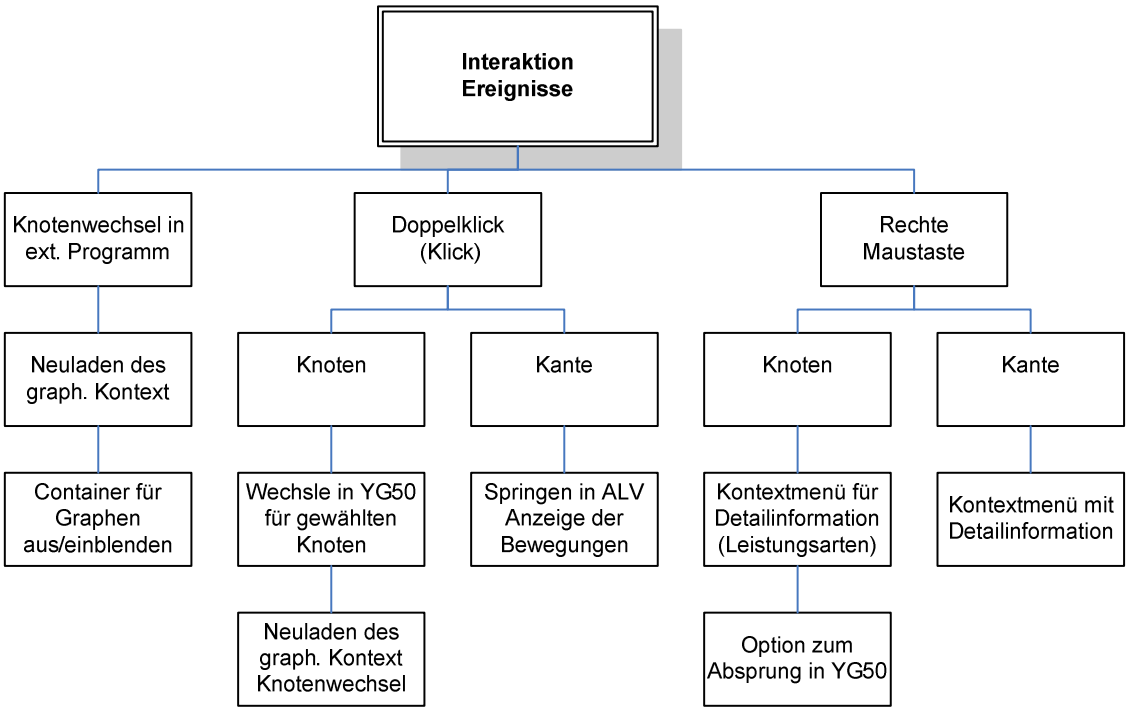


Abbildung 20 Anforderungen für die Ereignisverarbeitung

Im nächsten Abschnitt sollen die einzelnen Punkte der Abbildung näher beschrieben werden (s. Tabelle 5).

Tabelle 5 Anforderungen an die Ereignisverarbeitung

Feature	Beschreibung
Knotenwechsel Container aus- und einblenden	- Keine eigenständige Visualisierung sondern Einbettung in Rahmenapplikation - Äußeren Kontextwechsel beachten (Zustandsänderung in Rahmenapplikation) - Schnittstelle für Bekanntmachung eines neuen Kontexts vorsehen - Neuen Knoten immer in der Mitte platzieren - Visualisierung kann durch einen Button im Status der Rahmenapplikation gestartet werden
Doppelklick/ Klick auf Knoten	- Bei einem Klick soll der gewählte Knoten zentriert werden - Bei einem Doppelklick wird der gewählte Knoten zum neuen Ausgangspunkt. -> Graph wird neu geladen (aus Puffer) - Anpassen der Objekte an neue Situation
Doppelklick/ Klick auf Kante	- Vorsehen eines Absprungs in eine Detaildarstellung z.B. ALV Browser beim Doppelklick auf Kante - Bei einem einfachen Klick soll die entsprechende Kante fokussiert werden - Anpassen der weiteren Objekte an neue Situation (Komponenten ein- /ausblenden)

Feature	Beschreibung
Rechte Maustaste auf Knoten	- Anzeigen eines eigenen Kontextmenüs für Knoten - Absprung in Anzeigefunktion vorsehen - Menüpunkt zum Zentrieren des Knoten - Menüpunkt zum Anzeigen der Leistungsarten für CI - Vorsehen eines eigenen Graphen zur Anzeige von Leistungsarten (Prüfen geeigneter Modelle) - Vorsehen von Schnittstellen für weitere Verzweigung - (Ausblick) Menü anbieten um Knoten zu bearbeiten, z.B. einzelne FI bearbeiten, Knoten löschen, Knoten umhängen …
Rechte Maustaste auf Kante	- Vorsehen eines eigenen Kontextmenüs - Anzeigen aller Bewegungen für die Knotenbeziehung - Absprung in Detailanzeige z.B. ALV Browser - Menüpunkt anbieten um Kante zu zentrieren - (Ausblick) Optionen zum Bearbeiten der Kante anbieten, z.B. Kante löschen, Menge ändern …

Um die einzelnen Punkte näher zu erläutern, findet sich im Anhang eine erweiterte Beschreibung der o. g. funktionalen Anforderungen bzw. Funktionen.

3.5.2 Nicht-funktionale Anforderungen im Detail

Im Kapitel 3.1.2 wurde vorgestellt, was unter nicht-funktionalen Anforderungen zu verstehen ist. An dieser Stelle wurden ebenfalls drei Funktionsgruppen für nicht-funktionale Anforderungen definiert. Im folgenden Abschnitt soll auf diese drei Gruppen näher eingegangen werden.

Anforderungen an Technologien

Es existiert eine hohe Bandbreite an Werkzeugen, mit denen das graphische Modell einer Wertschöpfungskette angezeigt werden kann. Im konkreten Falle soll jedoch eine in SAP integrierte Lösung angestrebt werden, was die Auswahl erheblich einschränken sollte. In diesem Abschnitt sollen die einzelnen technischen Aspekte vorgestellt werden, die in der folgenden Übersicht abgebildet sind (s. Abbildung 21).

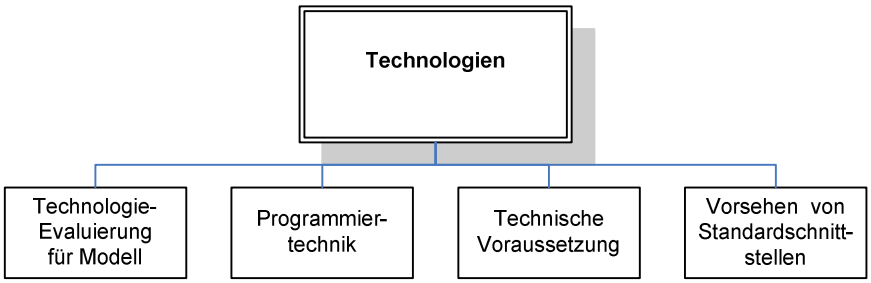


Abbildung 21 Anforderungen im Bereich Technologie

Die einzelnen Punkte der Anforderungsgruppe Technologien werden in der folgenden Tabelle näher beschrieben (s. Tabelle 6).

Tabelle 6 Nicht-funktionale Anforderung Technologien

Feature	Beschreibung
Evaluierung geeigneter Techniken	- Visualisierung sollte innerhalb eines SAP GUI laufen können - Unterstützung gängiger R/3 Basis-Releases 4.6C und 4.7 - Keine besonderen Rahmenbedingungen für Betrieb erforderlich - Ggf. Implementierung von Testprogrammen - Abwägung zusätzlich erforderlicher Werkzeuge - Auswahl einer „State of the Art“ Technologie
Programmier-technik	- Weitestgehend ABAP und ABAP OO benutzen - Objektorientiert programmieren - Auswahl gängiger Design-Patterns - Zentrale Speicherung der Objekte zur besseren Wiederverwendbarkeit.
Technische Voraussetzung	- Welche Voraussetzungen müssen auf dem Kundensystem erfüllt sein - Besonderheiten der verwendeten Technologie - Systemvoraussetzung - Prüfung auf Eingriffe in SAP Standard (Customizing, User-Exits)
Vorsehen von Standard-schnittstellen	- Implementieren von Standardschnittstellen, die von jeder Rahmenapplikation bedient werden können - Applikationsspezifische Klassen abstrakt vordefinieren und für konkrete Applikation implementieren - Includes für wieder verwendbaren Code erstellen

Anforderungen an die Datenbeschaffung

Dieser Abschnitt soll die Anforderungsgruppe Datenbeschaffung und Datenquellen näher vorstellen. Die folgende Grafik (s. Abbildung 22) zeigt die einzelnen Punkte in einer Übersicht.

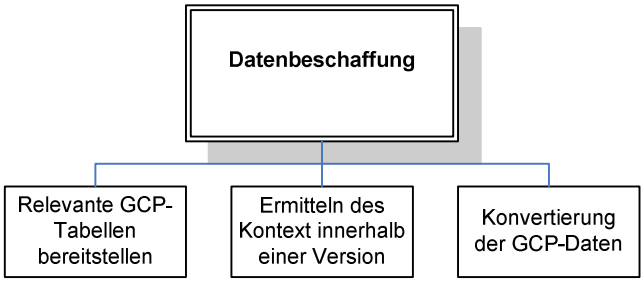


Abbildung 22 Anforderungen an die Datenbeschaffung

Im folgenden Abschnitt sollen die einzelnen Punkte näher vorgestellt werden, die in der Datenbeschaffung beachtet werden müssen (s. Tabelle 7).

Tabelle 7 Nicht-funktionale Anforderungen Datenbeschaffung

Feature	Beschreibung
Bereitstellung der GCP Quelldaten	- Konsistenz der GCP Quelldaten sicherstellen - Visualisierung auf Basis von FI, Kanten- und Knotensätzen - GCP Aggregation und Sequence müssen für die anzuzeigende Version gelaufen sein - Lesen der Quelldaten mit möglichst performantem DB Zugriff - Vermeiden von teuren Datenbankzugriffen
Ermitteln des passenden Kontexts	- Vorsehen eines performanten Algorithmus zur Abgrenzung eines Auszugs aus der gesamten Wertschöpfungskette - Keine Risse und Lücken in der Wertschöpfungskette vermeiden bzw. aufzeigen
Konvertierung der GCP Daten	- Automatische Konvertierung in Datenformat der Visualisierungskomponente. - Prüfung auf Wiederverwendbarkeit der vorhandenen Konvertierungsfunktionen - Arbeiten mit einer einheitlichen und verkürzten Datenstruktur (Nur benötigte Felder in den Hauptspeicher laden)

Anforderungen an Darstellung und Performance

Nachdem in den funktionalen Anforderungen festgelegt wurde, welche Informationen zu den Knoten und Kanten angezeigt werden sollen, soll es in diesem Abschnitt um die Art der Darstellung gehen. Die folgende Abbildung (s. Abbildung 22) stellt die einzelnen Anforderungen dar, die nun beschrieben werden.

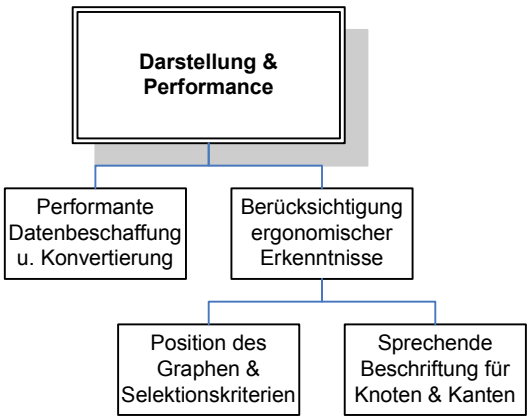


Abbildung 23 Anforderungen an Darstellung und Performance

Im folgenden Abschnitt sollen die einzelnen Anforderungen im Bereich Performance und Darstellung näher vorgestellt werden (s. Tabelle 8).

Tabelle 8 Nicht-funktionale Anforderungen Darstellung & Performance

Feature	Beschreibung
Performance der graphischen Darstellung	- Schneller Aufbau des Graphen nach dem Kontextwechsel - Schnelle Konvertierung der GCP Quelldaten - Keine Überlastung von Ressourcen und Hauptspeicher - Nur relevante Informationen verarbeiten
Berücksichti-gung	- Graphische Aufbereitung sollte im „SAP Stil“ erfolgen - Farben aus SAP GUI und SAP Design Spezifikationen verwenden

bildschirm-ergonomischer Daten	- Anordnen der Controls und Eingabefelder nach Richtlinien der Bildschirmergonomie - Vergleich mit aktuellen „EnjoySAP“ Transaktionen z.B. ME23N - Standard Schriftgrößen verwenden und lesefreundliche Farben verwenden - Optimale Position des Navigationsfensters feststellen - Der Benutzer soll Positionen der Controls manuell anpassen können. - Beschriftung der Knoten und Kanten mit sprechendem Namen Beachten von Conversion-Exits
Optimale Reaktion bei Kontextwechsel	- Bei Wechsel der Planversion werden alle Daten und auch die Puffertabelle neu geladen. - Überlagerung der alten Anzeigefunktion (virtueller neuer Modus), um Rücksprünge möglich zu machen. - Vor- und Rückwärtsnavigation ermöglichen (Buttons in Toolbar)

3.6 Benutzerschnittstellen

Die einzige für den Benutzer sichtbare Schnittstelle bietet das Fenster zur Navigation durch die Wertschöpfungskette. An dieser Stelle werden vom Benutzer alle nötigen Eingaben vorgenommen. Das Fenster zur Navigation soll in drei Abschnitte aufgeteilt werden. Ein Abschnitt dient der Eingabe der zu zeigenden Daten aus GCP, während der nächste Abschnitt den Layout Einstellungen dient. Den letzten und größten Teil des Navigationsfensters bildet die Darstellung der Wertschöpfungskette (s. Abbildung 24). Die Parameter für Daten und Layout sollen hierbei erst nach Knopfdruck erscheinen. Die Applikationen, für die eine Navigationsfunktion angedacht ist, müssen einen Button zum Anzeigen der Leiste im Bildschirmstatus vorsehen.

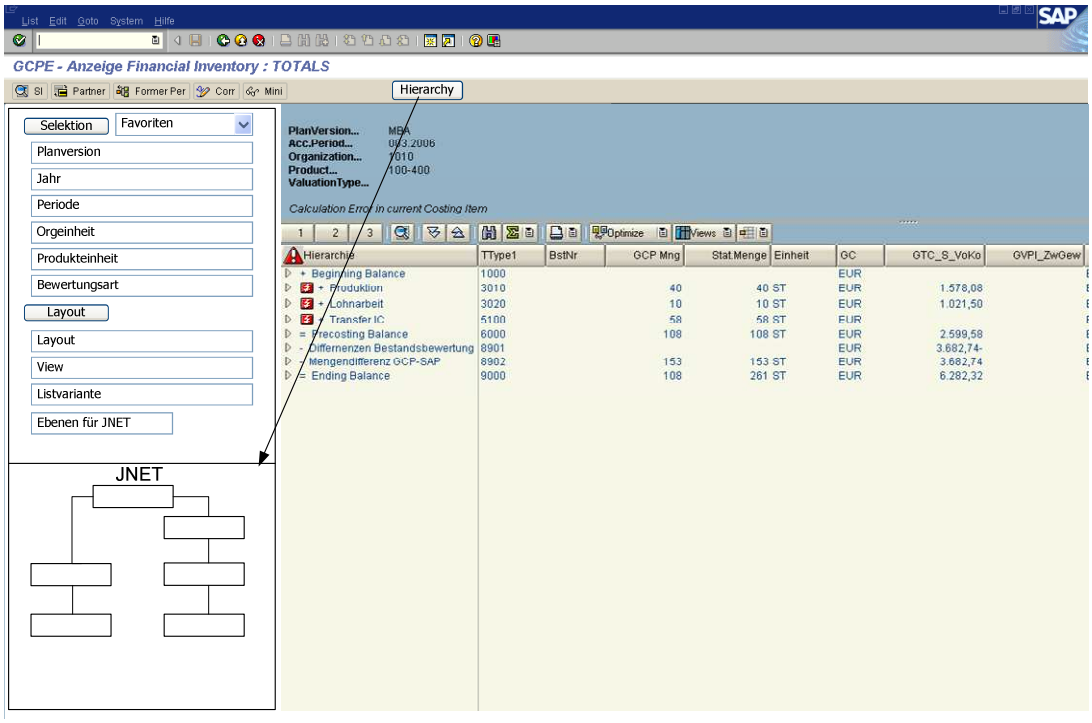


Abbildung 24 GCP Anzeigefunktion mit aktivierter Navigation

Für weitere Funktionen wie die Anzeige von Favoriten oder die Einrichtung von Verwendungsnachweisen sollen ebenso Bildschirmelemente vorgesehen werden. Die Favoriten könnten in einer Listbox erscheinen.

4 Technische Evaluation

In diesem Kapitel sollen einige Techniken vorgestellt und abgewogen werden, die zur Visualisierung von Wertschöpfungsketten innerhalb von SAP in Frage kommen. Es sollten für das vorliegende Projekt ausschließlich SAP eigene Techniken zum Einsatz kommen, um den Support und die Wartung gewährleisten zu können. Zudem sollte es möglich sein, den Wertschöpfungsgraphen innerhalb eines SAP GUI anzuzeigen.

4.1 Technologien

In diesem Abschnitt sollen zunächst alle in Frage kommenden Techniken vorgestellt werden. Dies sind zum einen die Techniken, die in den Standard integriert wurden und genau für einen Betrieb innerhalb des SAP GUI vorgesehen sind. Zum anderen gibt es neben diesen Standardtechniken noch externe Werkzeuge zur Visualisierung verschiedener Graphen. In den Vorgesprächen zum vorliegenden Projekt wurden einige Techniken vorgestellt und diskutiert. Hierbei fielen viele Möglichkeiten schnell unter den Tisch und sollen an dieser Stelle nicht näher betrachtet werden.

Alle Funktionen, die eine Abbildung von Grafiken innerhalb eines R/3 Systems ermöglichen, sind in der Komponente BC-FES-GRA zusammengefasst. Dieses Paket beschreibt alle grafischen Modelle, die zur Verfügung stehen. Die beiden folgenden Unterkapitel beschreiben zwei Technologien aus diesem Paket, die eine Untermenge abbilden.

4.1.1 Business Graphics Framework

Das SAP GFW (Business Graphics Framework) stellt grundsätzlich einige Standardklassen bereit, die es einem Anwender erlauben, bestimmte Graphen in seinen ABAP Report, d.h. sein Programm, einzubinden. Das GFW beinhaltet die Darstellung von Balken und Kuchendiagrammen sowie den Aufbau von hierarchischen Graphen, wie im vorliegenden Fall eine Wertschöpfungskette. Motivation dieses Frameworks ist die Verbesserung der Präsentation externer und interner Daten auf dem SAP GUI. Das GFW bietet hierfür die folgenden Funktionalitäten.[25]

- Das GFW beinhaltet eine MVC (Model View Controller) Pattern. Hierbei können bestimmte Grafiken den gleichen Datencontainer benutzen. Ändern sich die Daten dort, hat dies Auswirkungen auf alle abhängigen Views. Den Controller des Frameworks bildet die umgebende Applikation der Grafik. Diese übernimmt die Verarbeitung von Benutzereingaben und Ereignissen. Im Zuge der Verarbeitung letztgenannter übernimmt die Applikation als Controller auch die Modifikation des Datencontainers.

- Durch eine einheitliche Datenbasis ist es möglich die einzelnen Grafiken gegenseitig auszutauschen. Hierbei werden die Objekte im Rahmen neuer SAP Releases implizit aktualisiert.

- Da alle Grafiken im Standard SAP GUI dargestellt werden, sind keine zusätzlichen Installationen erforderlich. Da der SAP GUI für fast jede Plattform verfügbar ist,

[25] SAP AG (2001), S. 11.

besteht eine Plattformunabhängigkeit. Es ist jedoch darauf zu achten, dass entweder eine ActiveX Komponente oder eine JRE installiert ist.

- Das GFW steht in Form von Standardklassen zur Verfügung und ist somit bereits in ABAP OO implementiert. Durch die Unterstützung objektorientierter Eigenschaften steht eine komfortable Programmierschnittstelle bereit. Weiterhin ist das GFW in das Standard SAP Customizing integriert. Die verschiedenen Grafiken können benutzerspezifisch angepasst werden.

Die folgende Abbildung (s. Abbildung 25) zeigt eine Grafik zur Darstellung von Hierarchien. Genau dieser hierarchische Graph bezeichnet das passende Mittel, um globale Wertschöpfungsketten darstellen zu können.

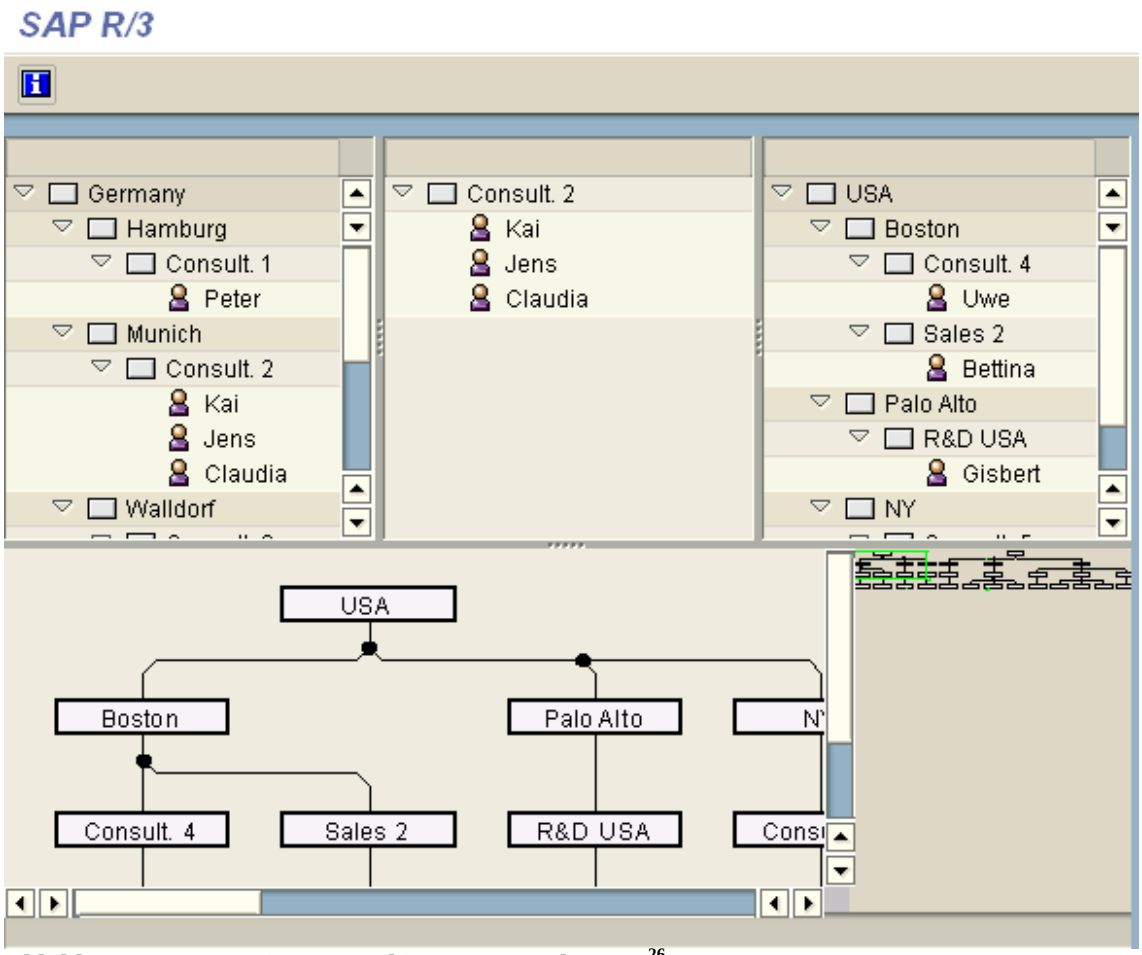


Abbildung 25 SAP Business Graphics Framework Demo[26]

Die Abbildung zeigt zunächst einzelne Verzeichnisbäume mit Hierarchien. Alle Hierarchien werden im unteren Teil der Abbildung als Organigramm dargestellt, der links eine Zoom- und Navigationshilfe bereitstellt. An diesem Beispiel ist auch die gemeinsame Nutzung des gleichen Datencontainers für alle angezeigten Objekte zu erkennen.

[26] SAP AG (2001), S. 10.

4.1.2 ABAP Graphics Development Network

Das ABAP Graphics Development Network stellt eine weitere Möglichkeit dar, Hierarchien innerhalb von SAP R3 Applikationen darzustellen. Die Funktionen dieser Modelle sind hierbei umfassender als im GFW selbst. Wobei das Framework selbst älter ist als das GFW. Der Graph wird in einem eigenen Fenster dargestellt. Dieses Fenster bietet sehr viele Möglichkeiten das Layout und die Daten selbst zu bearbeiten. Die SAP stellt für das ABAP Development Network einige Demo Programme zur Verfügung, die mit der Transaktion SE83, SAP Reuse Library getestet werden können. Vom technischen Aufbau ist das ABAP Development Network sehr ähnlich zum Business Graphik Framework. So nutzt das GFW einige Funktionen dieses Frameworks gleichzeitig mit.

Die folgende Abbildung (s. Abbildung 26) zeigt eines dieser Programme, wobei diese schon das hohe Alter der Technologie verdeutlicht.

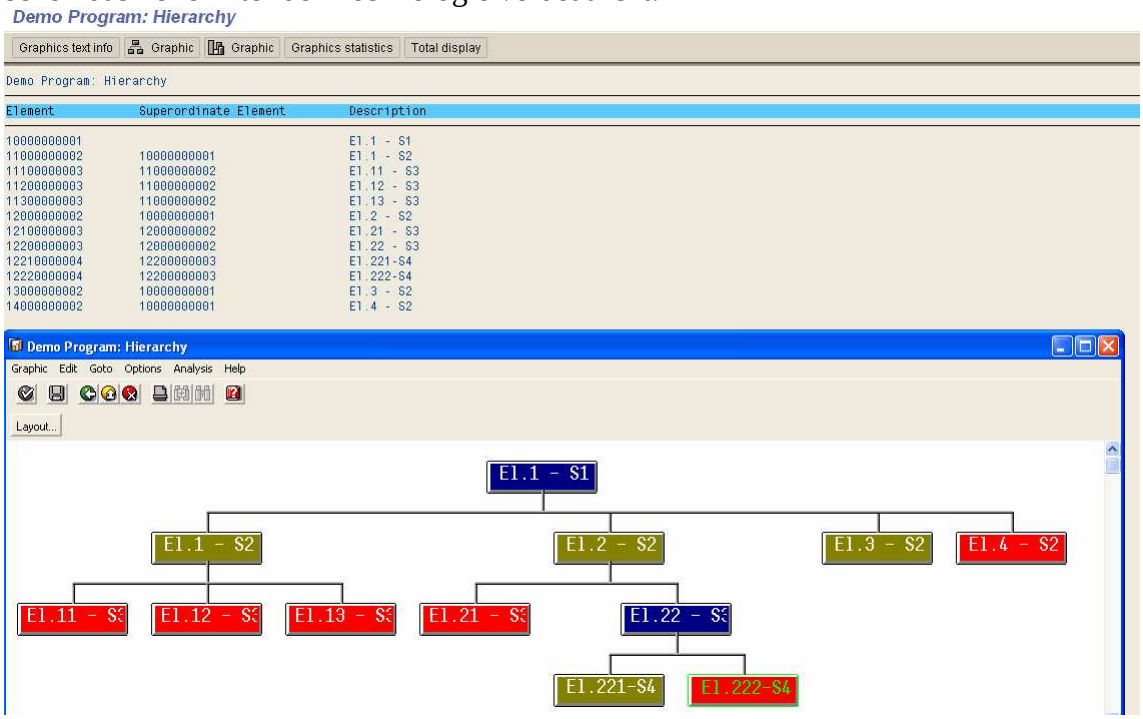


Abbildung 26 Grafik des ABAP Graphics Development Framework

Die Grafik zeigt einen SAP Report, der aus simplen Daten eine beispielhafte Hierarchiegrafik aufbaut. Diese wird in einem eigenen Bildschirm angezeigt. Das Design erreicht aber hierbei nicht die Ansprüche einer modernen Visualisierung von Wertschöpfungsketten.

4.1.3 JNET / JGANT

JNet und JGANT beschreiben ein Framework zur Darstellung verschiedenster Diagrammtypen auf allen denkbaren Plattformen. Im Gegensatz zu den vorangegangenen Technologien ist diese nicht in R/3 integriert, sondern es bedarf einer separaten Installation. JNet und JGANT fassen im Prinzip alle bisherigen grafischen Funktionen der SAP in einem neuen Paket zusammen, das auch außerhalb eines R/3 Systems zur Verfügung steht. Neben der Zusammenfassung wurden neue und nützliche Funktionen ergänzt und das Design komplett überarbeitet. JNet bietet die Möglichkeit sehr viele Modelle aus dem SAP Umfeld darzustellen. Grundsätzlich kann alles

dargestellt werden, dass aus Knoten besteht, die über Kanten verbunden werden sollen. Hierbei reicht die Auswahl von Organigrammen aus der Personalwirtschaft bis hin zu Modellen zur Automation von Kampagnen in CRM Modulen.[27] JGANT hingegen ist auf die Darstellung von GANT Diagrammen aus dem Projektmanagement spezialisiert und soll daher nicht mehr näher betrachtet werden.

In den neueren SAP Applikationen wie dem Netweaver und der mySAP Suite kommen die JNET Modelle häufig zum Einsatz. Dies erfolgt z.B. in Bereichen von Webapplikationen mit Business Server Pages, in welchen die JNet Bibliotheken als Taglibs bereits hinterlegt sind. Durch die Bereitstellung über Taglibs ist eine explizite Installation von JNet nicht mehr notwendig. Neben der Integration in Business Server Pages kann JNet auch ganz eigenständig ohne SAP betrieben werden. Hierfür ist eine eigenständige Applikation vorgesehen. Weiterhin kann JNet innerhalb von WebDynpro Anwendungen, Java-Applets, Java-Servlets und des SAP GUI, durch ein HTML-Control zum Einsatz kommen.

Die folgende Abbildung (s. Abbildung 27) zeigt ein Beispieldiagramm, das mit JNET erstellt wurde.

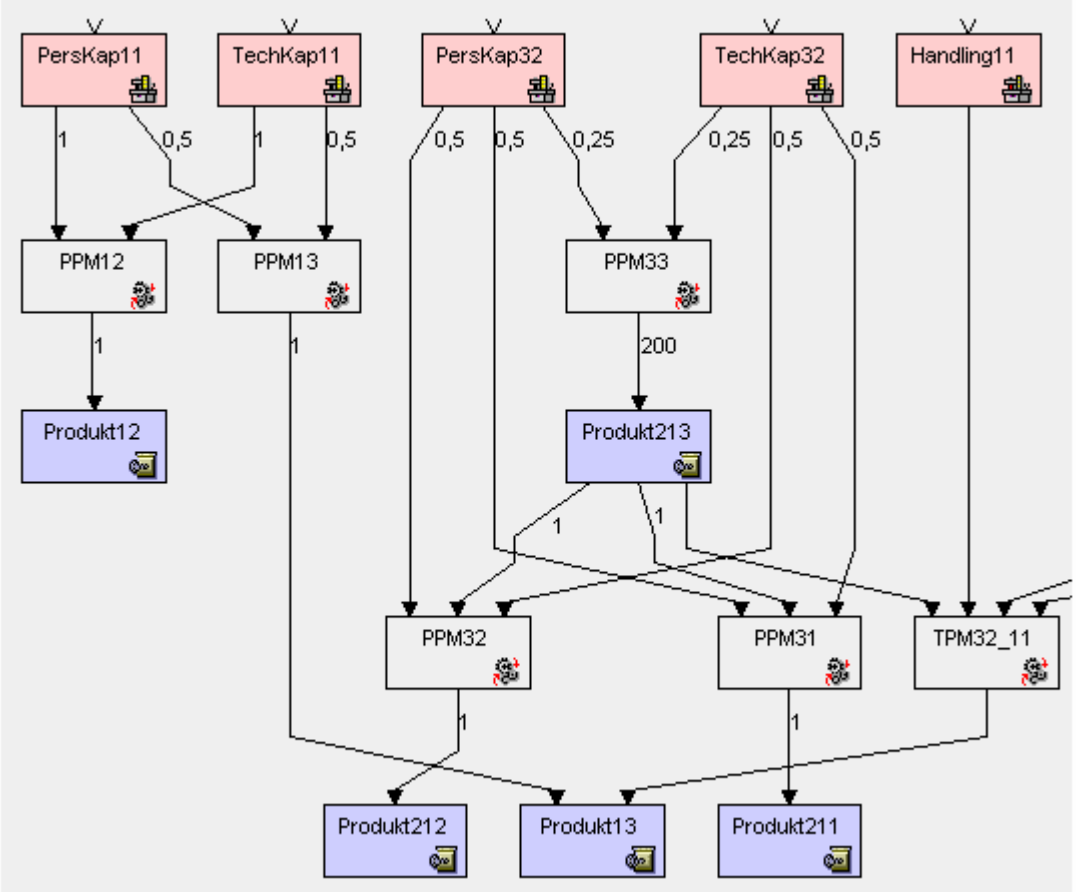


Abbildung 27 APO / PPM Diagramm in JNet[28]

[27] Vgl. https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/lw/uuid/f010ec31-9658-2910-3c83-c6e62904eceb eingesehen am 19.09.2007.
[28] Ebd.

Die Überarbeitung von Design und Elementen des Diagramms ist deutlich zu erkennen. Das hier gewählte Diagramm soll auch als Ausgangsbasis zur Anzeige der globalen Wertschöpfungsketten dienen. Das SAP Modul APO ist Bestandteil der MySAP SCM Suite. Es soll die Anwender bei Entscheidungen der Material Logistik unterstützen und übernimmt Aufgaben in der Bedarfs- und Produktionsplanung.[29] Da für die Planung von Primär- und Sekundärbedarf die Produktkalkulation TopDown gerechnet werden muss, stellt dies ebenso eine geeignete Methode dar, eine globale Wertschöpfungskette zu visualisieren. Natürlich müssen hierbei einige Änderungen in der Struktur getroffen werden, aber die Elemente bleiben dieselben. Die komplexe Typdefinition von JNet erlaubt es für jeden Knoten, eine separate Darstellung zu definieren und verschiedene Icons in den Elementen abzubilden.

Technisch ist JNet eine in Java geschriebene Applikation. Die Datenbasis für jedes Modell bilden speziell aufgebaute XML-Dokumente. Diese werden vom JNET Parser verarbeitet und die entsprechenden graphischen Elemente aufgebaut. Je nach Einstellung können die Elemente nun auf dem Bildschirm angezeigt, ausgewählt und bearbeitet werden. Ist eine Bearbeitung möglich, wird das XML-Dokument, welches die Daten enthält, direkt modifiziert. Die folgende Tabelle (s. Tabelle 9) stellt kurz die drei wichtigsten Elemente vor, auf denen JNet basiert.

Tabelle 9 Basisdaten eines JNet Diagramms[30]

Modelldaten (Data File)	Dieser Teil muss für jedes JNet Modell vorhanden sein. Die einzelnen Elemente des Graphen werden in diesem Abschnitt definiert. Die Datendefinition beginnt mit dem XML Element <SAPJNetData><Graph>. Vornehmlich werden Knoten (nodes), Kanten (edges) und Beschriftungen (labels) definiert, die dabei auf Typen aus dem Type Repository verweisen können. Wird kein Typ angegeben, erfolgt die Anzeige der Elemente sehr unübersichtlich und generisch.
Typdefinition (Type Repository)	In diesem Teil des XML-Dokuments können die Typen für die einzelnen Knoten, Kanten, Label etc. definiert werden. Die zentrale Definition von Typen hat den Vorteil, dass der Datenteil von unnötigen Formatoptionen befreit wird und keine redundanten Angaben gemacht werden müssen. Ein Typ eines Elements kann über das Attribut „type=“ angesprochen werden. Sollten im Datenteil Angaben zum Format eines Elements gemacht werden, übersteuert dies die Definition im Type Repository. Das Type Repository wird über das XML Element <TypeRepository> eingeleitet.
Benutzerschnittstelle (User Interface)	Die Benutzerschnittstelle definiert alle Attribute der Oberfläche des JNet Fensters. Diese besteht aus den Bereichen Statusleiste, Toolbar und Arbeitsbereich. Für diese Elemente können hiermit bestimmte Eigenschaften wie Größe und Sichtbarkeit definiert werden. Weiterhin ist es möglich, Attribute zu bestimmen, die für das ganze

[29] Vgl. Tempelmaier, Horst (2003), S. 377.
[30] Vgl. https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/lw/uuid/f010ec31-9658-2910-3c83-c6e62904eceb eingesehen am 19.09.2007.

	Modell gültig sind. Dies ist z.B. die Angabe eines initialen Zoomfaktors oder das Anlegen benutzerspez. Kommandos. Die Benutzerschnittstelle wird mit dem Attribut <UserInterface> angegeben.

Diese XML Basiselemente können entweder in einer XML Datei angegeben werden oder jeweils getrennt stehen. Es ist hierbei auch vom jeweiligen Modell abhängig, ob es neben den Modelldaten noch eine Typdefinition oder eine Benutzerschnittstelle gibt.

Bei den Recherchen im Vorfeld dieser Arbeit war JNet als die Zukunftslösung der SAP im Bereich von Visualisierungen zu identifizieren. Diese Aussage wurde auch in einigen Telefongesprächen mit der SAP AG verdeutlicht.

4.2 Analyse und Abwägung

In diesem Abschnitt sollen die drei vorgestellten Techniken den Anforderungen gegenübergestellt werden um zu prüfen, welche Technik zu wählen ist. Da alle Techniken in einem gewissen Maße aufeinander aufbauen, kann man generell nicht von Vor- und Nachteilen sprechen.

Für die beiden ersten Technologien, die rein in SAP R/3 zur Verfügung stehen, könnte es zu Problemen kommen, falls ein Kunde die Visualisierungskomponente in SAP BW betreiben möchte. Die notwendigen Standardklassen sind nicht verfügbar. Weiterhin ist es denkbar, dass ein Kunde seine Wertschöpfungskette ganz außerhalb eines SAP Systems betrachten möchte, z.B. als JPEG oder PNG Grafik. Auch dies würde zu Problemen führen, da im Standard keine Methode vorgesehen ist, die ein Diagramm in eine Bilddatei umwandeln kann.

Einen deutlichen Vorteil haben die beiden ersten Technologien in Sachen Customizing. Die notwendigen Einstellungen für die Anzeige der Modelle können im Rahmen des SAP Standardcustomizings vorgenommen werden. Hierbei ist es auch möglich benutzerspezifische Angaben zu machen.

An dieser Stelle muss für JNet ggf. ein eigenes Customizing entwickelt werden, das an die jeweiligen Wirtsysteme angepasst werden kann.

Die folgende Tabelle (Tabelle 10) stellt die Technologien den jeweils wichtigsten Anforderungen gegenüber und beurteilt hierbei die jeweilige Eignung. Die einzelnen Punkte werden durch eine Skala von – nach ++ bewertet.

Tabelle 10 Gegenüberstellung Anforderung - Technologien

Anforderung	Business Graphics	ABAP graph. Dev.	JNet
Hierarchische Darstellung von Knoten und Kanten	(+) Hierarchie-darstellung in Form von Organigrammen möglich	(+) Viele versch. Modelle zur Darstellung von Hierarchien mit Abhängigkeiten	(++) Spezieller gerichteter Graph verfügbar, der zur Anzeige von Wertschöpfungsketten vorgesehen ist (s.

Anforderung	Business Graphics	ABAP graph. Dev.	JNet
	Aussehen der Beschriftungen nur bedingt anpassbar. Kanten-beschriftungen nicht möglich.	Die graphischen Elemente sind jedoch sehr statisch. Keine gerichteten Verbindungen möglich. Die Beschriftung der Kanten ist ebenso sehr statisch.	Abbildung 27) Produkthierarchien. Verschiedene Verbindungstypen, Farben und Beschriftungen.
Einheitliche Darstellung bei gleichen Optionen	(--) Nur sehr schwer möglich verschiedene Typen von Knoten zu unterscheiden.	(-) Eingeschränkte Möglichkeit Designs in Knoten zu ändern bzw. unterschiedliche Strukturen abzubilden.	(+) JNet kann in einem Modell unterschiedliche Knoten darstellen. Eigenschaften können geprüft werden und jeder Knoten individuell angepasst werden.
Knotenwechsel und Refresh	(+) Bei einem Knotenwechsel wird ein neuer Graph erstellt. Für diesen muss die Datenbasis neu erstellt werden.	(+) Bei jedem Knotenwechsel muss der Datencontainer neu erstellt werden. Typen und Benutzerattribute müssen neu gesetzt werden.	(+) Bei einem Knotenwechsel kann die Datenstruktur für JNet dynamisch angepasst werden, da diese aus getrennten XML Dateien besteht, an die neue Daten angehängt werden können. Der Refresh führt zu einem erneuten Parsen der Datenbasis.
Inhaltlicher Zoom	(-) Liegt in der Verantwortung der Bereitstellung der Daten. Daten eines hinzugefügten Levels können nicht angehängt werden.	(-) Das inhaltliche Zoom wird bereits bei der Erstellung des Datencontainers berücksichtigt. Kein Anhängen von Daten möglich, sondern Neuaufbau des Containers.	(+) Das inhaltliche Zoom wird auch bei JNet bereits durch das XML-Dokument mit den Daten bereitgestellt. Man kann jedoch bestimmte Elemente ein- und ausblenden. Weiterhin können neue Level an das Dokument angehängt werden.
Visueller Zoom	(+) Alle drei Technologien verfügen über ein kleines Zoomfenster, um den Ausschnitt des Graphen zu bestimmen, der auf dem Bildschirm angezeigt werden soll. JNet bietet im Applet selbst		

Anforderung	Business Graphics	ABAP graph. Dev.	JNet
	die Möglichkeit in das Modell hinein- bzw. herauszuzoomen. Dabei können bestimmte Elemente abhängig vom aktuellen Zoom ein- und ausgeblendet werden.		
Ereignisbehandlung	(+) Ereignisse werden in einer separaten Klasse oder der umgebenden Applikation (Controller) verarbeitet. Nur eingeschränkter Set an Ereignissen.	(++) Da JNet in Java oder einem Java Applet läuft, werden auch dort die Ereignisse erzeugt und weitergeleitet. Über eine JavaScript Schnittstelle können die Ereignisse an eine äußere Applikation weitergegeben und verarbeitet werden. Viele vordefinierte Ereignisse sind vorhanden. Definition eigener Ereignisse ist möglich.	
Anforderungen im Ausblick	(-) Keine direkte Möglichkeit, die Daten direkt in den Knoten zu ändern. Es ist jedoch möglich, die Struktur des Graphen anzupassen. Hinzufügen von Knoten und Kanten ist möglich.	(+) Struktur der Grafik ist änderbar. Die Richtung der Kette kann bestimmt werden. Änderungen an den Knoten selbst sind nur eingeschränkt möglich.	(++) In JNet können neue Modelle von Grund auf definiert werden. Die Struktur wird hierbei im XML-Dokument abgelegt. Es ist auch möglich die Knoten und Kantenbeschriftungen zu bearbeiten. Die Kanten zwischen Knoten können leicht geändert werden. Über Parameter kann gesteuert werden, welche Elemente anpassbar sein sollen und welche nicht. XML Daten müssen später in GCP Daten umgewandelt werden.
Nicht-Funktionale Anforderungen	(++) Grundlegende Implementierung mit ABAP OO Controls. Klassen stehen global zur Verfügung. Sehr gute Performance, da in SAP GUI integriert, und der Datencontainer im DDIC definiert ist.		(-) Performance reicht nicht an die beiden anderen Technologien heran, da Daten in XML konvertiert werden müssen. Aufruf des Java Applets kostet beim ersten Start mehr Zeit.

Die Tabelle zeigt, dass jede Technologie bestimmte Vor- und Nachteile aufweisen kann. Die Flexibilität von JNet sowie die Zukunftssicherheit durch eine unabhängige

Basisapplikation sind jedoch enorme Vorteile, die zu einer Auswahl dieser Technik führten.

Der zu entwickelnde Prototyp soll diese Technik verwenden und über selbst geschriebene Klassen mit einem R/3 System kommunizieren können. Dafür waren im Vorfeld zunächst einige Tests notwendig, um herauszufinden, ob JNet, im Zusammenhang mit R/3, die benötigten Anforderungen erfüllen kann. Nach den erfolgreichen Tests konnte mit dem Design der Applikation begonnen werden, welches im nächsten Kapitel vorgestellt werden soll.

5 Design und Modellierung

Nachdem nun die Anforderungen zusammengetragen wurden und die zu verwendende Technik evaluiert ist, beginnt das Design der Applikation. In diesem Kapitel soll zunächst ein Überblick über die Komponenten und Klassen der Applikation geschaffen werden. Im weiteren Verlauf wird auf die einzelnen Klassen im Detail eingegangen und deren Interaktion miteinander vorgestellt.

5.1 Annahmen

Die Grundvoraussetzung zur Visualisierung einer globalen Wertschöpfungskette bilden die richtigen Daten. In dieser Arbeit wird nicht näher auf die Kontrolle und Korrektur der Daten aus dem GCP eingegangen. Im Prototyp ist eine Prüfung der Daten nicht vorgesehen, daher wird von einem einwandfreien Datenbestand ausgegangen.

Damit JNet in der vorliegenden Form fehlerfrei arbeiten kann, müssen auf jedem lokalen Rechner bestimmte Voraussetzungen geschaffen werden. Neben der richtigen Einstellung der Verbindung zu einem SAP System (Firewalls, Router etc.), muss eine JRE installiert werden. Der Prototyp wird mit der JRE Version 1.4.2 getestet. Spätere Releases sollten jedoch ebenso funktionieren. Das JNet Applet bedient sich innerhalb des SAP GUI bestimmter Klassen, die auf jedem lokalen Rechner ebenso installiert werden müssen, da diese in der Standardinstallation nicht vorhanden sind. Die notwendigen Standardklassen wurden in drei „jar“ Archive zusammengefasst, die in den Ordner „`C:\Programme\SAP\JNet`“ kopiert werden müssen. Es handelt sich um die folgenden drei Pakete, die aus dem SAP Intranet geladen werden können.

- JNet.jar — JNet Basisfunktionen und Appletklasse
- JNetApps.jar — Erweiterte JNet Funktionen (Container, Hierarchie)
- y.jar — Layouter für Baum- und Hierarchiestrukturen

Nach der Installation dieser Klassen, die in einer externen JNet Anwendung bereits mit-installiert werden, sollten sich alle Modelle im SAP GUI anzeigen lassen. Es ist jedoch empfehlenswert, die Browsereinstellungen für ein Applet zu ändern. Damit der SAP GUI das JNet Applet in der installierten JRE ausführt und nicht in der Microsoft VM, wie es standardmäßig eingestellt ist, muss eine neue Umgebungsvariable in den erweiterten Einstellungen der Systemsteuerung definiert werden.

Umgebungsvariable	***Wert***
„`JAVA_PLUGIN_WEBCONTROL_ENABLE`“	„1“[31]

Da die meisten Kunden der IM&C noch mit einem älteren R/3 Release (4.6C) arbeiten, wird auch der Prototyp im Entwicklungssystem mit diesem Releasestand entwickelt. Eine Portierbarkeit in neuere Systeme sollte keine Probleme darstellen, muss aber getestet werden.

[31] Vgl. http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4525262 eingesehen am 20.11.2007

5.2 Programmkomponenten

In diesem Abschnitt soll der Leser einen Überblick über die geplanten Programmkomponenten des Prototyps erhalten. Die Visualisierungskomponente soll grundsätzlich aus drei Hauptbestandteilen bestehen, welche hierbei ein MVC Design Pattern abbilden. Bei einem solchen MVC Pattern wird die Anwendung nach diesem Beispiel geteilt und in die Komponenten Model, View und Controller zerlegt. Der View übernimmt in diesem Zusammenhang die Darstellung von Daten aus dem Model. Das Model enthält seinerseits die Kernfunktionalität und beschafft die Daten z.B. von der Datenbank. Der Controller bildet zusammen mit dem View die Benutzerschnittstelle und ist für die Eingaben des Benutzers und die Delegation der Ereignisse verantwortlich.[32]

Der Prototyp besteht vor allem aus der View- und Modellkomponente des Design Patterns, während die Logik des Controllers zum größten Teil der Applikation überlassen bleibt, in die die Visualisierung integriert werden soll. Dies ist notwendig, da jede Komponente unterschiedliche Anforderungen an die Ereignisverarbeitung haben kann. Für eine vereinfachte Implementierung der Controllerlogik in eine Wirtsapplikation sollen jedoch einige Standardincludes vorgesehen werden.

Die folgende Abbildung (s. Abbildung 28). zeigt ein Diagramm mit den Komponenten der Visualisierung.

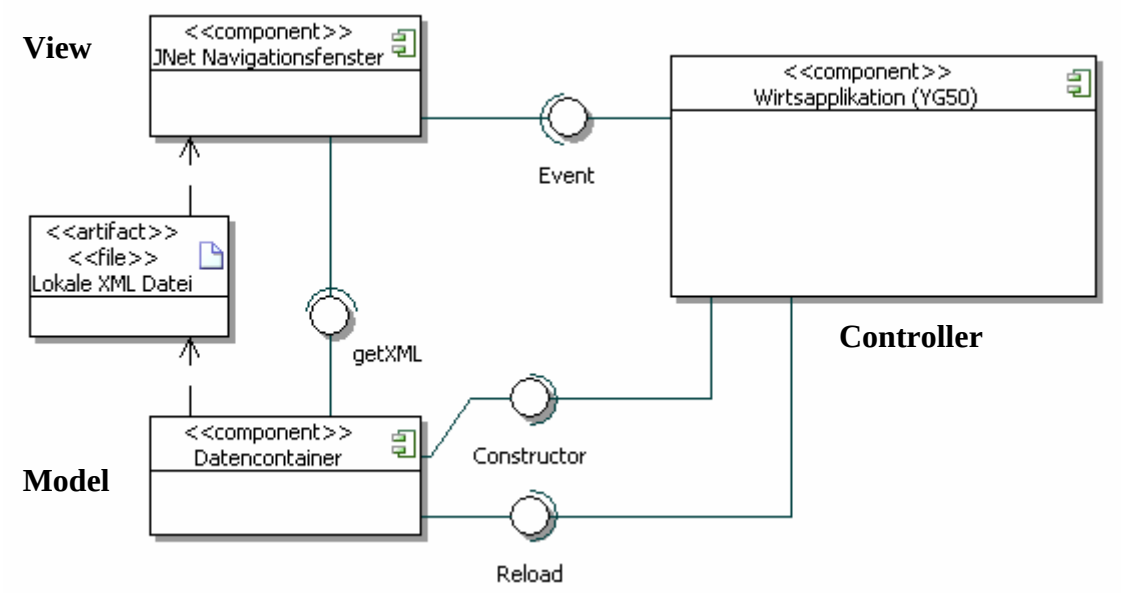


Abbildung 28 Komponentendiagramm Prototyp

Die Hauptkomponenten bilden die Wirtsapplikation(Controller) in Interaktion mit dem Datencontainer(Model) und dem Navigationsfenster(View). Die Komponenten sind über verschiedene Schnittstellen verbunden. Der Datencontainer bietet an dieser Stelle zwei Schnittstellen, durch die Methoden Constructor und Reload, um die Anzeigedaten zu erzeugen. Diese Schnittstellen werden jeweils von der Wirtsapplikation verwendet. Die Wirtsapplikation bietet ihrerseits durch die Integration eines Eventhandlers eine eigene Schnittstelle an. Hierbei fasst die Schnittstelle Event alle Methoden zur

[32] Buschmann, Frank et al. (1998), S.121.

Behandlung von Events zusammen. Die Schnittstelle wird durch die Verursachung von Events im Navigationsfenster angesprochen.
Neben diesen Abhängigkeiten bestehen bestimmte Verbindungen zwischen den Komponenten. Die Wirtsapplikation startet die Visualisierungsfunktion, wodurch der Datencontainer und das Navigationsfenster erzeugt werden. Die Benutzereingaben in der Anzeige lösen bestimmte Ereignisse aus, die zurück in der Wirtsapplikation verarbeitet werden müssen und deren Verhalten beeinflussen.

Zwischen dem Datencontainer und dem Navigationsfenster besteht keine direkte Schnittstelle. Der Datencontainer erstellt jedoch ein XML-Dokument, welches vom Navigationsfenster eingelesen und zur Anzeige gebracht wird.

Die einzelnen Komponenten bestehen ihrerseits aus mehreren Klassen, deren Aufbau und Zusammenspiel in den nächsten Abschnitten erklärt werden soll.

5.3 Klassenübersicht

In diesem Abschnitt sollen die geplanten Klassen vorgestellt werden. Details zu den einzelnen Klassen finden sich im jeweiligen Kapitel (s. 5.5). Es folgt zunächst eine Grafik (s. Abbildung 28) mit dem Klassenüberblick. An dieser Stelle sei weiterhin auf den Anhang verwiesen, welchem das komplette Klassendiagramm entnommen werden kann.

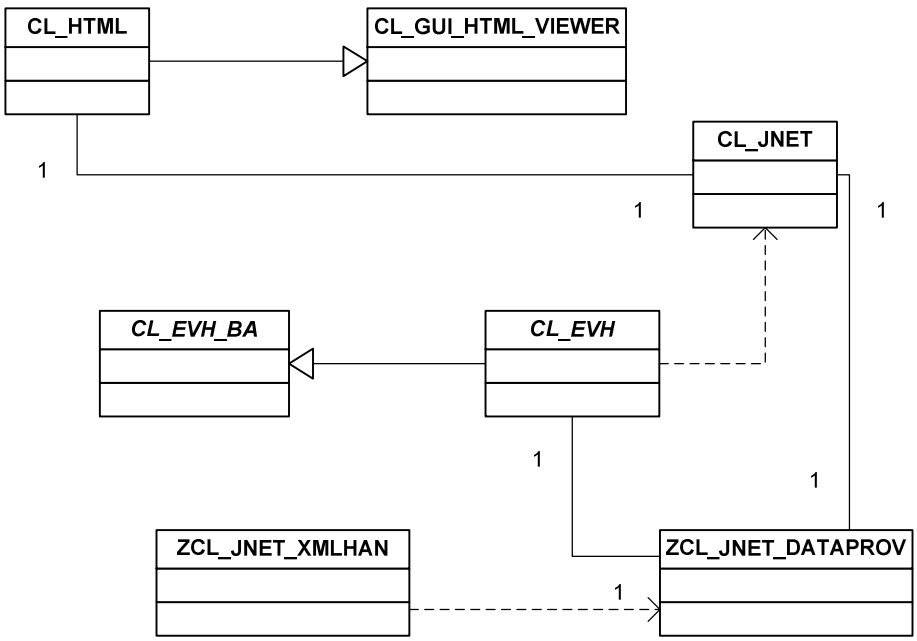


Abbildung 29 Klassenüberblick

Die Abbildung zeigt die geplanten Klassen, wobei die aus dem SAP System verfügbaren Standardklassen nicht mit dargestellt werden. Eine Ausnahme bildet die Klasse CL_GUI_HTML Viewer, da diese als Basis einer Vererbung dient.

Die beiden Klassen ZCL_JNET_XMLHAND und ZCL_JNET_DATAPROV bilden zusammen die Komponente für die Datenbasis (Model). Hierbei übernimmt die Klasse ZCL_JNET_DATAPROV die Beschaffung der Grunddaten aus der Datenbank. Weiterhin wird dort ein Pufferungsmechanismus erstellt, der für ein schnelleres Nachladen der Daten sorgen soll. Es wird also eine Datentabelle mit einer bestimmten Anzahl von Knoten gefüllt, die aber später nicht alle angezeigt werden. Für die Ermittlung der anzeigerelevanten Daten ist ein weiterer Algorithmus implementiert, der schließlich die Modelldaten zur Konvertierung an den XML-Handler übergibt. Ein Datenprovider hat immer genau einen XML-Handler, den er bei seiner Instantiierung erzeugt. Über einen Nachlademechanismus können schließlich neue Daten ermittelt werden, die im Optimalfall direkt aus dem Puffer geladen werden können.

Der XML-Handler ist seinerseits rein für das Erstellen und Bearbeiten eines intern gehaltenen XML-Dokuments verantwortlich. Dabei ist ein Teil des XML-Dokuments statisch und wird beim Erzeugen des Handlerobjekts automatisch angelegt. Ein zweiter Teil besteht aus den eigentlichen Modelldaten und wird bei jedem Neuladen überschrieben. Das erstellte XML-Dokument wird von der Klasse CL_JNET angefordert und als Datenbasis für das Java Applet verwendet. Der Benutzer kann über Parameter steuern, ob eine externe XML Datei erstellt werden soll. Zum Erzeugen einer solchen Datei integriert der XML-Handler bestimmte Methoden zum Parsen und Rendern des XML Objekts.

Die Klassen CL_JNET und CL_HTML bilden die eigentliche Benutzerschnittstelle der Applikation. Hier erfolgen alle notwendigen Schritte zur Erzeugung des Navigationsfensters. Die Klasse JNet muss jedoch auch den Datenprovider kennen, um einige Funktionen aus dem Datenprovider ansprechen zu können und die benötigten Daten zu laden. Die Klasse CL_JNET erzeugt zunächst ein HTML Control in dem an die Klasse übergebenen Containerelement. Bevor das HTML zur Anzeige kommt, müssen die Webseiten geladen werden, in denen das Applet ausgeführt wird. Die Klasse JNet ist weiterhin für die Weiterleitung der versch. Ereignisse an den Eventhandler (Controller) verantwortlich. In der Klasse CL_HTML wird von der Standardklasse für HTML Controls abgeleitet und diese um einige Methoden ergänzt, die nötig sind, um Scripts innerhalb einer Webseite, die im HTML Control angezeigt wird, auszuführen.

Die Klasse CL_EVH übernimmt im Zusammenspiel mit der Wirtsapplikation die Controller Komponente. Dabei wird der Event-Handler per Include in die Wirtsapplikation eingebunden. Die Events werden von der Klasse JNet je nach Typ an den Eventhandler delegiert, wobei Parameter mit übergeben werden können. Der Eventhandler steht in einer einfachen Verbindung mit dem Datenprovider, da manche Ereignisse Methoden des Datenproviders ansprechen. Dies ist z.B. der Fall, wenn neue Daten nachgeladen werden sollen oder ein Level ergänzt wird. Durch spezielle ABAP OO Anweisungen kennen sich die Klasse JNet als Erzeuger eines Events und die Klasse CL_EVH als Verarbeiter eines Events. Ohne die Klasse JNet würde kein Eventhandler gebraucht, wodurch die Abhängigkeit der Klasse CL_EVH zur Klasse JNet besteht.

Die Implementierung der einzelnen Methoden des Eventhandlers ist von der umgebenden Applikation abhängig. Eine abstrakte Basisklasse soll die Standardmethoden beschreiben, die Implementierung jedoch einer für jede Applikation abgeleiteten Instanz des Eventhandlers überlassen.

5.4 Schnittstellen

Entsprechend der in Kapitel 3 vorgenommenen Spezifikation soll die Komponente zur Visualisierung Standardschnittstellen zur Verfügung stellen, um die Einbindung in weitere Standard Applikationen zu vereinfachen. Die folgende Abbildung (s. Abbildung 30) soll die definierten Schnittstellenparameter verdeutlichen.

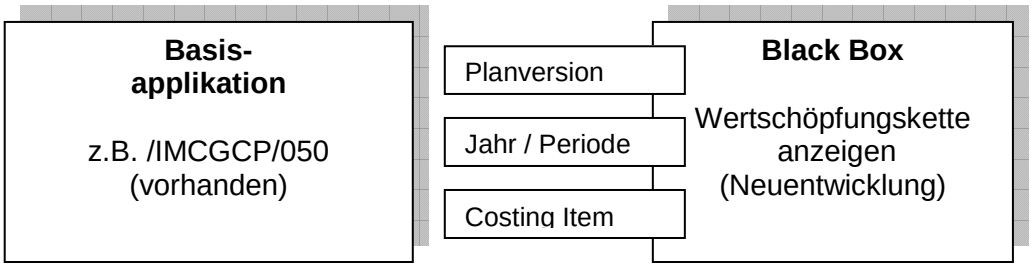


Abbildung 30 Black Box mit Schnittstellen

Die Grafik zeigt die "Black-Box" Sicht auf die Visualisierungskomponente mit deren entsprechenden Parametern. Diese setzen sich aus den Selektionskriterien für die zugrunde liegenden Daten zusammen. Die Schnittstellen des Model, die im Komponentendiagramm dargestellt wurden, benötigen diese Parameter. Die Planversion bestimmt zusammen mit Jahr und Periode für welchen GCP Lauf die Daten visualisiert werden sollen. Den wichtigsten Bestandteil der Schnittstelle bildet jedoch das Costing Item, welches gerade angezeigt wird bzw. welches als Ausgangspunkt für die Levels der Visualisierung dienen soll.

In diesem Zusammenhang selektiert der Datenprovider genau die gesamte Umgebung dieses CI in die Puffertabelle. Es werden danach n Stufen der Wertschöpfungskette nach oben und nach unten, ausgehend von diesem CI visualisiert und dieser Ausgangspunkt mit einer bestimmten Farbe selektiert. Die folgende Tabelle stellt die Schnittstellen zur Visualisierungskomponente dar und beschreibt diese kurz.

Tabelle 11 Schnittstellen der Visualisierungskomponente

Schnittstelle	Parameter	Beschreibung
Constructor	Planversion Jahr/ Periode Costing Item Level	Beim ersten Start der Visualisierung erhält der Datenprovider die Parameter für die Datenselektion. Diese werden über den Konstruktor gesetzt, der die weitere Verarbeitung startet. Die Planversion unterscheidet hierbei die verschiedenen Extraktionsläufe, die weiterhin auch durch Periode und Jahr eingeschränkt werden können. Das Costing Item beschreibt den Ausgangspunkt des Graphen und bildet einen speziellen Knoten ab. Der Level beschreibt wie viele Levels ausgehend vom Costing Item angezeigt werden sollen.
Reload	Costing Item	Die Reload Methode wird durch einen Navigation in der Wertschöpfungskette gestartet. Hierbei bleiben die Grunddaten wie Planversion und Periode gleich. Es ändert sich jedoch der

Schnittstelle	Parameter	Beschreibung
		Ausgangspunkt der, in Form eines Costing Item, an die Methode übergeben wird.
getXML	XML_Table	Liefert das im Datencontainer erzeugte XML-Dokument zurück, welches in eine Ausgabetabelle gerendert wurde.

Um eine reine Visualisierung anzubieten, reichen die oben beschriebenen Schnittstellen. Da aber neben der Anzeige auch bestimmte Reaktionen und Verarbeitungen in den Wirtsapplikationen stattfinden sollen, muss eine Schnittstelle von der Anzeige der Visualisierung zurück zur Wirtsapplikation geschaffen werden. Die folgende Tabelle stellt diese Schnittstellen vor:

Tabelle 12 Schnittstellen der Rahmenapplikation (Eventhandler)

Schnittstellen von der Visualisierung zurück in Basis Appl. / Event	
APPLET_EVENT	Event, das durch eine Benutzeraktion innerhalb des JNet Applets ausgelöst wird. Z.B. Knoten wurde ausgewählt
HTML_EVENT	Event, das durch eine Benutzeraktion im HTML Control erzeugt wird, in welchem das JNet Applet eingebettet ist. Z.B. Klick auf Hyperlink.
FUNCTION_SEL	Event, das durch das Betätigen eines Buttons in der Werkzeugleiste ausgelöst wird. Z.B. Vergrößern des JNet Fensters

Innerhalb der Wirtsapplikation muss ein Eventhandler durch einen Include eingefügt werden. Dieser fängt die einzelnen Ereignisse ab, die in der Benutzerschnittstelle erzeugt wurden. Die einzelnen Abarbeitungen der Ereignisse bleiben den verschiedenen Applikationen überlassen.

5.5 Klassen im Detail

Nachdem nun ein Überblick über die Klassen und Schnittstellen der Applikation geschaffen wurde, soll nun das Design der einzelnen Klassen selbst beschrieben werden. Hierbei wird zunächst die Definition der Klasse im Klassendiagramm dargestellt und danach ein Einblick in wichtige Methoden gegeben.

5.5.1 XML Data Provider (ZCL_JNET_DATAPROV)

Der Datenprovider ist eine der Basisklassen für die Visualisierung der globalen Wertschöpfungsketten. Hierbei ist dieser für die Beschaffung der Quelldaten von der Datenbank verantwortlich und enthält weiterhin Algorithmen zur Ermittlung der relevanten Daten in der Anzeige und zur Pufferung von Daten zur Verbesserung des Nachladeverhaltens. Die folgende Abbildung (s. Abbildung 31) zeigt die Definition dieser Klasse als Ausschnitt aus dem Klassendiagramm.

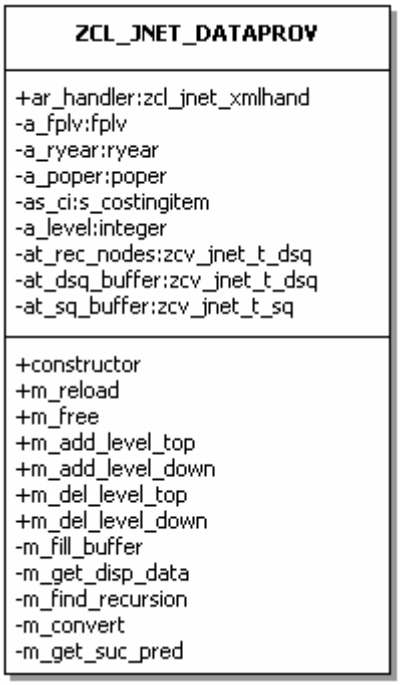


Abbildung 31 Klassendefinition Dataprovider (Detail)

Die folgende Tabelle stellt die im Klassendiagramm eingetragenen Attribute und Methoden der Klasse vor:

Tabelle 13 Beschreibung Klasse ZCL_JNET_DATAPROV

Klasse ZCL_JNET_DATAPROV	
Beschreibung:	Übernimmt die Beschaffung der Daten aus den GCP – Quelltabellen. Kommunikation mit XML-Handler
Attribute:	
ar_handler	Referenz auf Objekt der Klasse XML-Handler, um Modell dynamisch zu erstellen.
a_fplv, a_ryear, a_poper	Planversion, Jahr und Periode, um Quelldaten einzugrenzen
as_ci	Ausgangspunkt der Navigation (Costing Item)
a_level	Anzahl der Navigationsstufen von oben nach unten

Klasse ZCL_JNET_DATAPROV	
at_rec_nodes	Tabelle, die alle Knoten beinhaltet, die an einer Rekursion beteiligt sind. (Noch nicht implementiert)
at_dsq_buffer	Puffertabelle für Knoten des Graphen (Costing Items)
at_sq_buffer	Puffertabelle für Kanten des Graphen (Verbrauch, Transfer)
Methoden:	
constructor	Setzen der initialen Attribute und zentrale Steuerung bei erstem Start der Komponente. Fülle Puffertabelle, ermittle Kontext und starte XML-Konvertierung
m_reload	Setzt neuen Ausgangspunkt. Neuselektion des Kontext wird ausgeführt und der Datenteil des XML-Dokuments neu erstellt
m_free	Gibt Attribute und Tabellen wieder frei
m_add_level_top	Fügt dem aktuellen Modell einen Level nach oben hinzu, falls vorhanden
m_add_level_down	Fügt dem aktuellen Modell einen Level nach unten hinzu, falls vorhanden
m_del_level_top	Löscht den obersten Level der Wertschöpfungskette
m_del_level_down	Löscht den untersten Level der Wertschöpfungskette
m_fill_buffer	Füllt Puffertabelle (hierfür wird m_get_suc_pred rekursiv aufgerufen)
m_get_disp_data	Ermittelt aus der Puffertabelle den anzuzeigenden Kontext
m_find_recursion	Methode soll die an einer Rekursion beteiligten Knoten ermitteln. Algorithmus jedoch zu komplex, um dies zur Laufzeit durchzuführen. Daher nicht implementiert
m_convert	Konvertiert die Daten des Ermittelten Kontext in XML-Format (Bezieht sich auf Datenteil des XML-Dokuments)
m_get_suc_pred	Ermittelt für ein Costing Item alle direkten Vorgänger und Nachfolger

5.5.2 XML-Handler (ZCL_JNET_XMLHAND)

Der XML-Handler übernimmt alle Aufgaben im Bereich der Modifikation und des Aufbaus des XML-Dokuments, auf dem die aktuelle Anzeige in JNet basiert. Hierbei stehen auch Methoden zum Erzeugen einer Datei auf dem Applikationsserver und dem Parsen des Dokuments zur Verfügung. Über die Referenz des Datenproviders liefert der XML-Handler das in eine Charaktertabelle gerenderte Dokument an die JNet Anzeige. Die folgende Abbildung (s. Abbildung 32) zeigt zunächst die Definition der Klasse als Auszug aus dem Klassendiagramm.

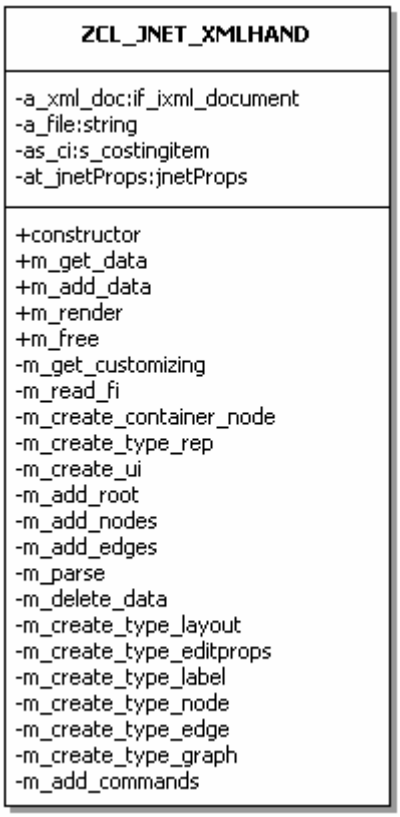


Abbildung 32 XML-Handler Definition (Detail)

Die folgende Tabelle stellt die im obigen Klassendiagramm eingetragenen Attribute und Methoden der Klasse genauer vor:

Tabelle 14 Beschreibung der Klasse ZCL_JNET_XMLHAND

Klasse ZCL_JNET_XMLHAND	
Beschreibung:	Klasse übernimmt den dynamischen Aufbau eines XML-Dokuments in Form eines DOM-Baum.
Attribute:	
a_xml_doc	Referenz auf das intern gehaltene XML-Objekt
a_file	Dateiname, falls eine externe Speicherung des XML gewünscht wird
as_ci	Ausgangspunkt der Navigation (Costing Item)
at_jnetprops	JNet-Customizingoptionen für aktuelles Profil
Methoden:	
constructor	Initiale Steuerung des XML-Dokuments. Erzeugen des XML-Objekts und Laden des TypeRepository und des UserInterface.
m_get_data	Zurückliefern des XML-Dokuments in Form einer internen Tabelle an den Datenprovider. (Konvertierung in binäre interne Tabelle)
m_free	Gibt Attribute und Tabellen wieder frei
m_add_data	Schreiben des dynamischen Datenteils in das XML-Dokument entsprechend der Daten aus dem Dataprovider

Klasse ZCL_JNET_XMLHAND	
m_render	Ermöglicht das Schreiben des erstellten XML-Dokuments in eine externe Datei
m_get_customizing	Ermitteln des JNet-Customizing für das aktuelle Profil
m_read_fi	Einlesen von der Financial Items, die zum entsprechenden Costing Item gehören
m_create_container_node	Erstellen eines Containerknoten für den Datenteil
m_create_type_rep	Erstellt das statische TypeRepository für das aktuelle Modell
m_create_ui	Erstellt das UserInterface für das aktuelle Modell
m_add_root	Fügt dem XML-Objekt ein Wurzelelement hinzu. Erste Operation mit DOM
m_add_nodes	Fügt dem Datenteil des XML-Dokuments alle ermittelten Knoten hinzu
m_add_edges	Fügt dem Datenteil des XML-Dokuments alle Verbindungen zwischen den Knoten hinzu
m_parse	Übernimmt das Parsing zum Einlesen eines geänderten XML-Dokuments (derzeit noch nicht implementiert)
m_delete_data	Löscht den nicht mehr benötigten Datenteil des XML-Dokuments, z.B. bei einem Kontextwechsel
m_create_type_layout	Erstellt das gewünschte Layout im TypeRepository
m_create_type_editprops	Erstellt die gewünschten Änderungseigenschaften für das Modell
m_create_type_label	Erstellt die verschiedenen Labeltypen- und Eigenschaften für das Modell
m_create_type_node	Erstellt die unterschiedlichen Typen für Knoten im Modell, im TypeRepository
m_create_type_edge	Erstellt die verschiedenen Typen für Kanten innerhalb des aktuellen Modells im TypeRepository
m_create_type_graph	Erstellt den Typ des Graphen im TypeRepository
m_add_commands	Fügt dem XML-Dokument Kommandos an den JNet-Interpreter hinzu, die nach dem Start der Applikation ausgeführt werden sollen

5.5.3 JNET Container (CL_JNET)

Die Klasse JNet beschreibt den wichtigsten Teil der Benutzerschnittstelle. Sie ist hauptsächlich für den Aufbau des JNet Fensters verantwortlich. Hierbei kann der Klasse ein beliebiger GUI Container (mr_parent) als Schnittstelle übergeben werden. Weiterhin übernimmt diese Klasse das Erstellen der lokalen temporären Dateien, aus dem SAP Web Repository und der XML Tabelle, die der Datenprovider liefert. Die folgende Abbildung (s. Abbildung 33) zeigt zunächst die Definition der Klasse CL_JNET als Auszug aus dem Klassendiagramm.

Abbildung 33 Definition Klasse JNet (Detail)

Die folgende Tabelle soll die im zuvor dargerstellten Klassendiagramm eingetragenen Attribute und Methoden näher beschreiben:

Tabelle 15 Beschreibung der Klasse CL_JNET

Klasse CL_JNET	
Beschreibung:	Übernimmt die Steuerung und Initialisierung des JNet Navigationsfensters in Interaktion mit der Wirtsapplikation
Attribute:	
m_dataprov	Referenz auf den Datenprovider für Datenerzeugung
m_url_frame	URL der HTML Datei mit dem Frameset. Frameset verhindert die Anzeige doppelter Scrollleisten
m_url_page	URL der temporär erstellten HTML-Datei
m_url_xml	URL der temporär erstellten XML-Datei
m_ev_frame	Eventparameter für aktuellen Frame
m_ev_action	Eventparameter für ausgelöste Aktion
m_ev_getData	Inhalt der „Get“-Variablen aus Event – Formulardaten
mt_ev_postData	Inhalt der „Post“-Variablen aus Event – Formulardaten
mt_ev_query	Variable mit dem Inhalt der aktuellen Anfrage aus Event
mr_html	Referenz auf Objekt der Klasse CL_HTML. Dies ist das HTML-Control zur Anzeige des Applets
mr_parent	Referenz auf den übergeordneten Container, in welchem die Navigation untergebracht wird. Z.B. Docking Container
mr_split	Referenz auf Splitter-Container, der in Docking Container integriert wurde

Klasse CL_JNET	
mr_xml_object	Referenz auf Standard XML-Objekt, um generische XML Tabelle umzuwandeln
m_guid	Eindeutiger String, um die jeweils neu erstellten temporären Dateien zu benamen
m_temp_dir	String enthält das temporäre Verzeichnis der aktuellen Plattform
mr_xml	Generische XML-Tabelle (XML-Objekt) wurde in eine binäre Tabelle übersetzt, um diese auf den Rechner zu speichern
m_delete_dl	Indikator der bestimmt, ob vorhandene (alte) temporäre Dateien gelöscht werden sollen
Methoden:	
free	Setzt alle intern gehaltenen Tabellen und Attributwerte zurück
reloadGraph	Aktuell angezeigte Grafik wird in dieser Methode neu erstellt und zur Anzeige gebracht (Kontextwechsel erfolgt)
constructor	Übernimmt das Anlegen der nötigen Objekte und übernimmt den Initialstart der Anzeige mit dem Triggern der Datenerstellung
load_xml	Ermittelt das im XML-Handler erstellte Dokument und setzt dieses in eine Binärtabelle um. Es erfolgt der Download in das lokale Verzeichnis des aktuellen Rechners
load_html_file	Laden und Erstellen der temporären HTML-Datei, die das Applet zur Anzeige des Wertschöpfungsgraphen enthält
on_sap_event	Leitet in JNet erzeugte Events an den Eventhandler weiter
button_pressed	Leitet in Splitter Container bzw. Toolbar erzeugtes Event an den Eventhandler weiter

5.5.4 Event-Handler (CL_EVH)

Die Klasse CL_EVH ist Bestandteil der Komponente Controller. Hierbei muss diese Klasse applikationsspezifisch implementiert werden. Somit sollte als Standard eine abstrakte Basisklasse mit den Methodenrümpfen zur Verfügung stehen, von der für die jeweilige Applikation abgeleitet wird. Im weiteren Verlauf wird diese Klasse als Implementierung in der GCP Anzeigefunktion, wie diese im Prototyp zum Einsatz kommt, vorgestellt. Die folgende Abbildung zeigt zunächst die Definition des Eventhandlers als Ausschnitt aus dem Klassendiagramm (s. Abbildung 34).

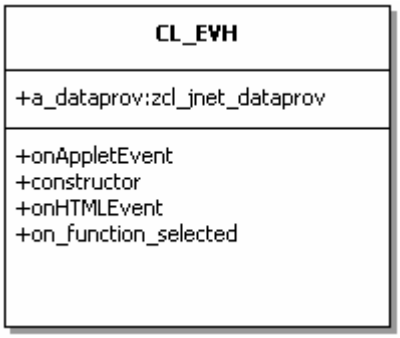


Abbildung 34 Definition Klasse CL_EVH (Detail)

Die folgende Tabelle soll die o. g. Klassendefinition mit geeigneten Beschreibungen für die Attribute und Methoden ergänzen:

Tabelle 16 Beschreibung der Klasse CL_EVH

Klasse CL_EVH	
Beschreibung:	Übernimmt als Erweiterung der Wirtsapplikation die Verarbeitung der empfangenen Ereignisse. Die einzelnen Ereignisverarbeitungen können je nach Rahmenapplikation erweitert werden.
Attribute:	
a_dataprov	Referenz auf den Datenprovider, um Modifikationen an den vorliegenden Daten vornehmen zu können.
Methoden:	
onAppletEvent	Diese Methode fängt alle Events ab, die in dem JNet Applet erzeugt wurden. Die Unterscheidung erfolgt über Event-Parameter
constructor	Übernimmt die Initialisierung der Klasse. Der Datenprovider wird dem Attribut zugeordnet
onHTMLEvent	Verarbeitet weitere Ereignisse aus der HTML-Seite. Im Moment ist kein entsprechendes Ereignis definiert
on_function_selected	Fängt die Ereignisse aus der Werkzeugleiste des JNet-Navigationsfensters ab und verarbeitet diese. Beispiel: Ausklappen des Navigationsfensters

5.6 Dynamisches Verhalten

In diesem Abschnitt soll das dynamische Verhalten der Visualisierung vorgenommen werden. Hierbei müssen zunächst verschiedene Zustände bzw. Szenarien unterschieden werden. Im Anhang dieser Arbeit findet sich ein Sequenzdiagramm, welches die Interaktion der verschiedenen Klassen beim ersten Programmstart vorstellt und beschreibt.

Den Start der Interaktion löst hierbei der Benutzer aus, der den Button zur Anzeige der Wertschöpfungskette drückt. Durch dieses Ereignis wird die JNet Steuerung gestartet, die ihrerseits die einzelnen benötigten Klassen anlegt. Nachdem die Klassen angelegt wurden, lädt die JNet Klasse die dynamisch erzeugte HTML-Seite und zeigt diese im entsprechenden Control an. In einem vorherigen Schritt wurde bei der Erzeugung des

Datenproviders das XML-Dokument erstellt und konvertiert. Im letzten Schritt werden die Inhalte der Controls auf dem Bildschirm ausgegeben und die Navigation durch das Setzen des Eventhandlers gestartet.

Als exemplarisches Beispiel für das dynamische Verhalten der Visualisierung soll im folgenden Abschnitt die Ereignisbehandlung für einen Doppelklick dienen. Zunächst soll in der folgenden Abbildung ein UML Sequenzdiagramm dargestellt werden, welches die Interaktion zwischen den Klassen näher beschreiben soll (s. Abbildung 35).

Die folgenden Sequenzdiagramme finden sich im Anhang:
- Systemverhalten bei Start der Visualisierung
- Ereignisverarbeitung bei Doppelklick auf Knoten

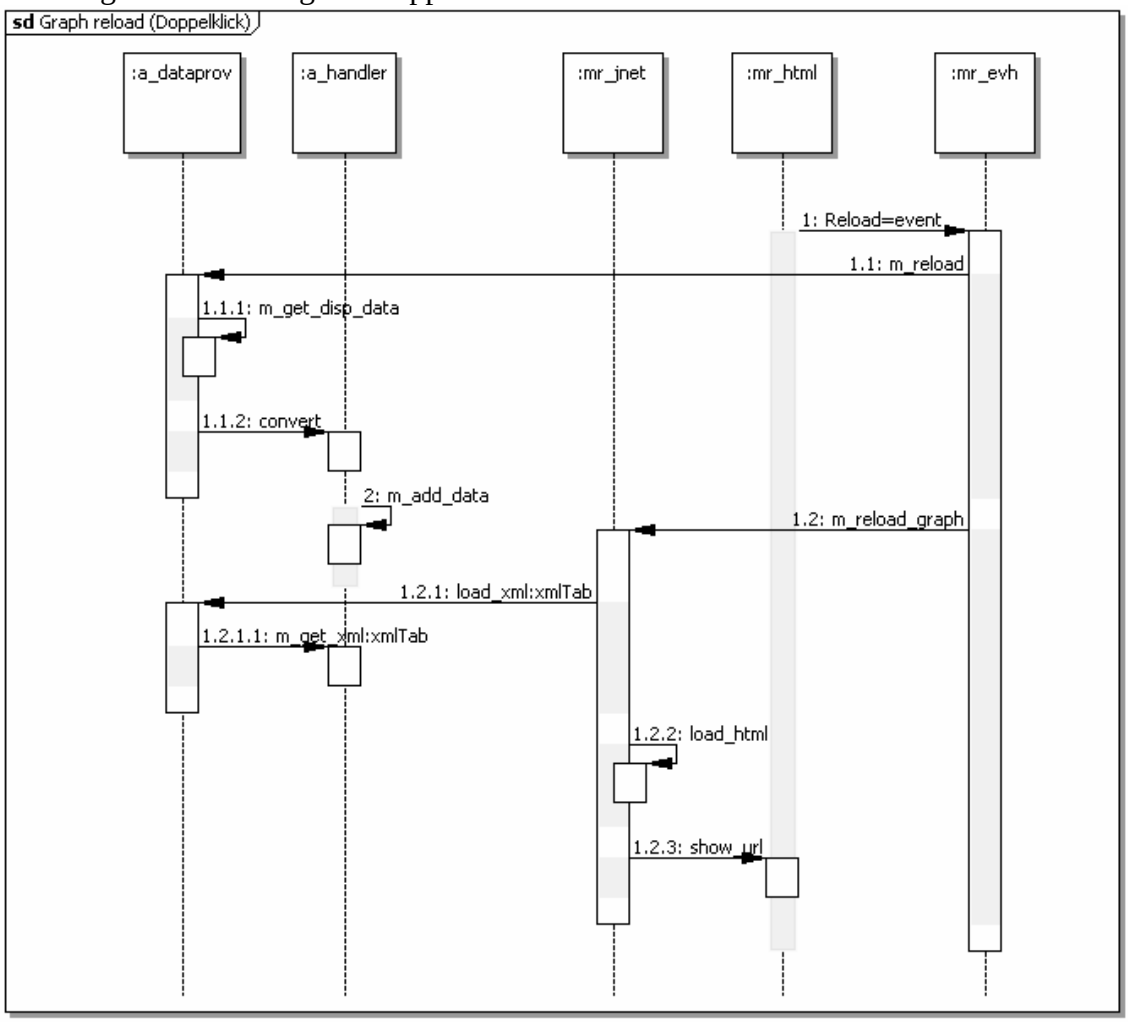


Abbildung 35 Sequenzdiagramm bei Doppelklick

Im Diagramm sind oben die beteiligten Objekte dargestellt. Diese wurden beim ersten Start der Visualisierung erzeugt (Vgl. Sequenzdiagramm Start Visualisierung im Anhang C). Die Events werden innerhalb des HTML-Controls durch das JNet Applet erzeugt und an den Eventhandler delegiert. Dieser wertet das Ereignis aus und übernimmt die entsprechende Verarbeitung. Im Falle des Doppelklicks ändert sich der Ausgangspunkt der Visualisierung. Es müssen somit neue Daten in den Graphen nachgeladen werden können. Der Eventhandler ruft also die Methode m_reload des Dataproviders auf und teilt diesem den neuen Ausgangspunkt in Form eines CI mit.

5.7 Designalternativen

Das Design einer solchen Applikation besteht nicht aus einer perfekten Lösung. Es gibt vielmehr einige Ansätze zur Diskussion. Im folgenden Abschnitt soll eine der möglichen Designalternativen vorgestellt werden, wobei es sicherlich noch einige andere Lösungen geben kann.

Im Zuge der Anzeige von Modellen muss JNet eine XML Datei als Datenbasis erhalten, welche danach geparst wird. Es soll in der Applikation jeweils eine Puffertabelle für Knoten und Kanten geben. In dieser Puffertabelle werden die anzuzeigenden Knoten und Kanten markiert und gemerkt. In einem weiteren Schritt werden die anzuzeigenden Daten in einer Übergabetabelle gespeichert und an den XML-Handler zur Konvertierung übergeben. Durch den Aufruf des Konverters bleibt zwar der statische Teil des XML-Dokuments, UserInterface und TypeRepository, erhalten, der dynamische Teil mit den bisher angezeigten Knoten und Kanten wird jedoch komplett gelöscht und neu erstellt.

An dieser Stelle wäre die Alternative, die Daten im XML-Dokument nicht komplett zu löschen, sondern mit den Inhalten der Übergabetabelle abzugleichen. Diese Maßnahme erspart es ggf. einen Teil der Quelldaten neu zu konvertieren, da der bereits erstellte Abschnitt im XML-Dokument wieder verwendet werden kann.

Die Vorgehensweise eines Abgleichs der Übergabetabelle mit dem vorhandenen XML Dokument erfordert jedoch das Parsen bzw. Einlesen des Dokuments nach bestimmten Parametern. Das XML-Dokument geht beim Suchen von Elementen sequentiell vor, da es, im Gegensatz zu internen Tabellen, über keinen optimierten Zugriff, z. B. in Form eines Schlüssels, verfügt. Somit muss das Dokument für jeden Knoten solange durchlaufen werden, bis der entsprechende Eintrag gefunden wird.

Es kann also anstelle einer Performanceverbesserung zu einer Verschlechterung kommen, falls das XML-Dokument jedes Mal durchlaufen werden muss, um den Eintrag zu finden. Um die beiden Alternativen zu vergleichen, wurde ein kleines Testprogramm erstellt, welches ermitteln sollte, welche Alternative ein schnelleres Ergebnis liefert.

Der Vergleich dieser beiden Vorgehensweisen lieferte ein eindeutiges Ergebnis. Der Aufbau eines komplett neuen Datenteils ist deutlich schneller, als das Suchen nach einzelnen Elementen. Dies lässt sich durch die ungünstige sequentielle Suche im XML-Dokument erklären. Der Test wurde weiterhin mit verschiedenen Datenmengen durchgeführt. Im Anhang findet sich das ausführliche Testprotokoll mit dem Vergleich der beiden Alternativen.

6 Realisierungsansätze

Nach der Erstellung der Anforderungsanalyse und dem entsprechenden Design der Software kann mit der Entwicklung begonnen werden. Zunächst sollen die einzelnen Schritte der Realisierung vorgestellt werden. Danach soll auf die Besonderheiten der einzelnen Komponenten eingegangen werden. Am Ende der Realisierung soll die Applikation, insbesondere im Zusammenhang mit den primären Use-Cases und Anforderungen, getestet werden.

6.1 Entwicklung eines objektorientierten Prototypen

Am Beginn der Entwicklung stellte sich die Frage, wie die JNet Applikation in den SAP GUI eingebunden werden kann. Hierzu wurde zunächst ein kleines Testprogramm erstellt, welches ermöglichte, ein beispielhaftes JNET Modell in einem HTML Control darzustellen. Dieses Testprogramm wurde bereits in der Technologieevaluation entwickelt, um festzustellen, ob JNet überhaupt in einem SAP GUI laufen konnte. In weiteren Schritten wurde dieses Testprogramm durch Features wie z.B. eine rudimentäre Ereignisverarbeitung ergänzt. Dies diente weiteren Tests, ob JNet den Anforderungen aus Kapitel 3 entsprechen konnte. Die folgende Abbildung (s. Abbildung 36) zeigt einen Screenshot des Testprogramms.

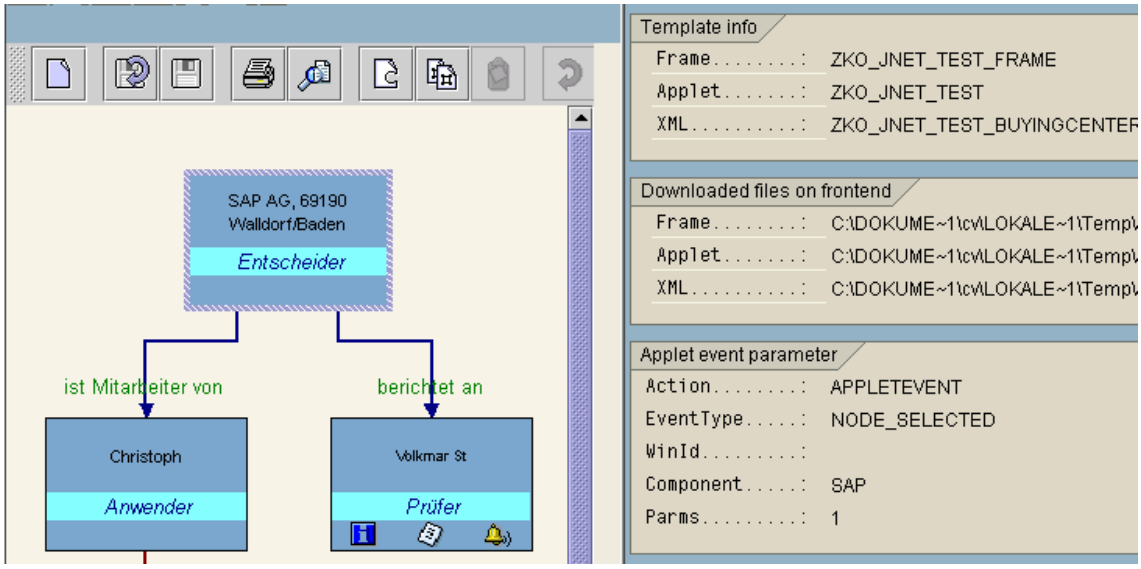


Abbildung 36 Screenshot Testprogramm im SAP GUI

Die Abbildung zeigt auf der linken Seite die Anzeige einer JNet Beispielanwendung. Auf der rechten Seite wird die entsprechende Ereignisverarbeitung dargestellt, die sich jedoch nur darauf beschränkt, die Art des Ereignisses mit den entsprechenden Parametern anzuzeigen.

Die Testapplikation besteht aus einem Selektionsbildschirm und einem Dynpro (Dynamisches Programm). Der Selektionsbildschirm bestimmt den Pfad der Beispieldatei für JNet. Das Dynpro enthält die Felder für die Parameter aus dem Ereignis (s. Abbildung 36 rechts). Im objektorientierten Kontext stellt ABAP eine Reihe von Standardcontrols zur Verfügung, die es erlauben, bestimmte andere Controls in sich einzubetten. Im Testprogramm wurde zunächst des Dynpro als Basiscontainer angelegt. In einem weiteren Schritt wurde ein Dockingcontainer erzeugt und an der

linken Seite des Bildschirms platziert. Der Dockingcontainer dient als Container für ein HTML-Control, welches ebenfalls benötigt wurde, um die HTML-Seite mit dem JNet Applet darstellen zu können. HTML-Controls ermöglichen es, Webseiten innerhalb des SAP GUI darzustellen. Die SAP verwendet diese Art von Controls sehr häufig bei neueren Transaktionen in R/3. Nach der Definition der einzelnen Container stellte eine Testklasse die korrekte Instantiierung der Container sicher. Nun musste eine Webseite erzeugt werden, die ein JAVA Applet beinhaltet. Dieses Applet musste die entsprechenden JAVA Archive von JNet als Codebase erhalten. Die Webseite wurde danach zentral auf dem Applikationsserver abgelegt. Das HTML-Control zog sich nach dem Programmstart die HTML-Datei und die Beispieldatei für ein JNet Modell. Die HTML-Seite wurde nun entsprechend im Control auf der linken Seite abgebildet. Schließlich musste noch eine Testklasse zur Ereignisverarbeitung erstellt werden. Für diese wurde der Eventhandler Mechanismus, den ABAP Objects zur Verfügung stellt, verwendet. Bei dieser Verarbeitung werden in den einzelnen Controls Ereignisse (Events) ausgelöst, die durch definierte Event-Handler-Methoden behandelt werden. Zuvor musste definiert werden, welche Ereignisse vom Control ausgelöst werden sollen und welche Methode das entsprechende Ereignis behandelt. Die Ereignisse können hierbei parametrisiert werden. Diese Technik ist vergleichbar mit den Event-Listenern aus der grafischen Programmierung von JAVA.

Diese Testapplikation diente als Basis der weiteren Entwicklungen, wobei manche Teile neu entwickelt und andere Teile übernommen werden konnten.

6.2 CL_JNET Containerfenster

Die Klasse CL_JNET übernimmt die Kontrolle der Anzeige. Diese Klasse wurde nicht im DDIC (Data Dictionary) angelegt, sondern in einem Programm Include definiert. Um eine JNet-Anzeige in ein Programm integrieren zu können, muss somit dieser Include in das Programm aufgenommen werden. Dieser Teil des Programms konnte von dem zuvor definierten Programm übernommen und erweitert werden. Zunächst wurde eine Möglichkeit geschaffen, die Daten für ein Modell über eine Schnittstelle von einer externen Komponente zu laden, wofür eine Erweiterung der Methode load_xml nötig war. In einem weiteren Schritt wurden das JNet Fenster durch eine Werkzeugleiste ergänzt. Um diese einbinden zu können, musste der Dockingcontainer durch einen SplitScreen-Container geteilt werden. Die folgende Abbildung soll den Zusammenhang der Container und Controls verdeutlichen.

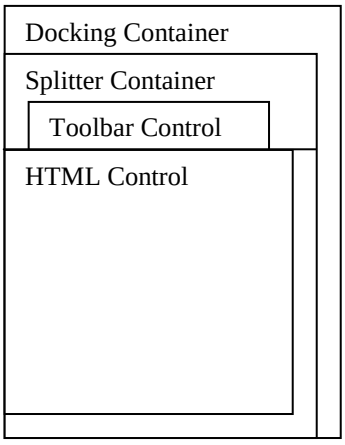


Abbildung 37 Beziehungen von Containern und Controls

Die Abbildung zeigt den Dockingcontainer als erste Ebene. Dieser Bereich wird durch einen Splitcontainer in zwei weitere Ebenen aufgeteilt, wobei eine Ebene für die Werkzeugleiste und die andere für das HTML-Control vorgesehen ist.
Durch die Erweiterungen in der Anzeige mussten auch die Eventhandler neu entwickelt werden. Zum einen wurde der Event-Handler für die Ereignisse aus dem HTML-Control erweitert, zum anderen ein neuer Handler für die Ereignisse aus der Toolbar erstellt. Die Events, die aus dem HTML-Control als SAPEVENT stammen, wurden nach HTML und Applet unterschieden. Hierbei ist jedoch zu beachten, dass die Behandlung der einzelnen Ereignisse applikationsabhängig ist. Im Kapitel 6.5 wird ein exemplarisches Beispiel vorgestellt, wie die Ereignisse in der CI Anzeigefunktion behandelt werden.

Ähnlich verhält es sich bei der Werkzeugleiste. Hier mussten zunächst die Buttons erstellt und mit Funktionscodes belegt werden. Nun musste ein Eventhandler erstellt werden, der die einzelnen Funktionscodes der Buttons unterscheiden konnte und entsprechende Funktionen auslöst. Auch hier ist es möglich, für jede Applikation eine individuelle Verarbeitung von Ereignissen zu definieren. Es wurde jedoch ein Set von Standardfunktionen zur Verfügung gestellt.

Innerhalb der Klasse CL_JNet wurde auch die Dateiverarbeitung entwickelt. Die statische HTML Beispieldatei wurde durch ein Template ersetzt, welches zur Laufzeit durch dynamische Inhalte ergänzt werden kann. Da die Daten nun dynamisch bereitgestellt werden sollten, wurde für jedes erstellte Modell eine eigene temporäre XML Datei erzeugt und auf dem lokalen Rechner abgelegt. Dies hatte den Vorteil, dass eine Vorwärts- bzw. Rückwärtsnavigation eingebaut werden konnte, für die kein weiteres Erstellen von XML Dateien mehr notwendig war. Beim Programmstart wurde ein Parameter vorgesehen, der die temporären Dateien des vorherigen Laufes löscht und somit ein Ansammeln von Daten vermeidet.

In den nächsten beiden Abschnitten (Kapitel 6.3 & 6.4) werden die beiden Programmkomponenten vorgestellt, die dem Anzeigefenster das dynamisch erzeugte XML-Dokument liefern.

6.3 XML Data Provider

Der XML Datenprovider kümmert sich um die korrekte Beschaffung der GCP-Daten und agiert als Schnittstelle zwischen der GCP und der JNet Visualisierung. Der folgende Abschnitt soll die einzelnen Schritte der Entwicklung dieser Komponente beschreiben.

6.3.1 Entwicklung der Komponente

Diese Komponente wurde als Klasse zentral im DDIC (Data Dictionary) des R/3 Systems angelegt. Dies bedeutet, dass Änderungen in dieser Komponente auf alle Applikationen, die eine Visualisierung verwenden, einen Einfluss haben. Weiterhin konnte durch die Verwendung des Class Builders von SAP eine schnellere Definition der Attribute und Methoden erfolgen.

Vor der Definition mussten jedoch noch einige einzelne Datenelemente und Strukturen im DDIC angelegt werden, da diese später von der Klasse verwendet wurden. Um keinen Platz im Hauptspeicher zu verschwenden, wurde eine verkürzte Struktur für die Knoten- und Kantentabelle definiert.

Für jede Änderung an Programmen oder DDIC-Elementen muss ein Workbench Transport angelegt werden. Dies ist erforderlich, da in der Praxis meist Entwicklungs- und Produktivsysteme getrennt sind. Die Transporte stellen sicher, dass alle Entwicklungen korrekt aus dem Entwicklungssystem in das Produktions- oder Konsolidierungssystem übernommen werden. Für die Entwicklungen der JNet-Visualisierung wurde ein eigener Transportauftrag eröffnet. Weiterhin wurde, um die Elemente der Visualisierung strikt von anderen GCP-Elementen zu trennen, eine eigene Entwicklungsklasse definiert. Entwicklungsklassen in SAP sind hierbei vergleichbar mit Paketen in Java, die logische Funktionsbereiche voneinander trennen.

Zunächst mussten die Schnittstellen definiert werden, die festlegen, für welche Datenbasis ein Modell erzeugt werden soll. Die Schnittstellen wurden, wie im Design angedacht, durch zwei Methoden angeboten. Im Falle des ersten Starts wird die Schnittstelle durch den Konstruktor abgebildet. Kommt es zu einer Navigation, dient die Methode m_reload als Schnittstelle.

Die folgende Abbildung zeigt einen Screenshot des SAP Class Builders mit einem Teil der Klassendefinition des Datenproviders (s. Abbildung 38).

Properties | Interfaces | Attributes | Methods | Events | Internal types | Aliases

Parameters | Exceptions | Filter

Methods	Level	Visi...	Mo...	C...	Description
CONSTRUCTOR	Insta...	Pub...	☐		Constructor
M_RELOAD	Insta...	Pub...	☐		Initialization of the data displayed in the model
M_FREE	Insta...	Pub...	☐		Free all resources
M_ADD_LEVEL_TOP	Insta...	Pub...	☐		Add one level to the top of the model / add receiver
M_ADD_LEVEL_DOWN	Insta...	Pub...	☐		Add one level to the bottom of the model / add sende
M_DEL_LEVEL_TOP	Insta...	Pub...	☐		Delete one level at top of the model
M_DEL_LEVEL_DOWN	Insta...	Pub...	☐		Delete one level at the bottom of the model
M_FILL_BUFFER	Insta...	Pri...	☐		Fill buffer table from database
M_GET_DISP_DATA	Insta...	Pri...	☐		Determine display data
M_CONVERT	Insta...	Pri...	☐		Convert the nodes and edges to xml data
M_FIND_RECURSION	Insta...	Pri...	☐		Find recursions over the entire supply chain.
M_GET_SUC_PRED	Insta...	Pri...	☐		Find succesors and predecesors for a specific node

Abbildung 38 Klassendefinition Dataprovider im SAP Class Builder

Innerhalb des Datenproviders wurde die geforderte Pufferung realisiert. Hierbei mussten zunächst alle Knoten der relevanten Stufen, + zwei Stufen als Puffer, eingelesen werden. Danach wurde ein Algorithmus eingebaut, der automatisch die für den Ausgangspunkt relevanten Knoten aus der Puffertabelle identifiziert und mit einem Anzeigeflag versieht.

In einem letzen Schritt wurden die anzeigerelevanten Daten an den XML Handler (s. Kapitel 6.4) übergeben, um ein dynamisches XML-Dokument zu erstellen.

6.3.2 Beschreibung des Ergebnisses

Der Datenprovider erstellt aus den Parametern einer Version zunächst eine Puffertabelle für alle Knoten und Kanten. In einem weiteren Schritt wird innerhalb der Puffertabelle der aktuelle Kontext der Anzeige ermittelt. Hierfür werden die direkten Nachfolger und Vorgänger eines Knotens mit einem rekursiven Algorithmus ermittelt.

Innerhalb der Klasse existieren zwei echte Datentabellen. Neben den echten Daten enthalten diese auch die Information, ob ein Knoten bzw. eine Kante angezeigt wird oder nicht. Dies hat den Vorteil, dass der Datenprovider immer weiß, welche Knoten gerade in der Anzeige sind und welche lediglich im Puffer verweilen. Dieses Prinzip ist Basis für den Algorithmus zur Ermittlung des nächsten und vorherigen Levels. Jeder Satz, der an dieser Stelle eine Markierung zur Anzeige trägt, wird in eine Übergabetabelle hin zum XML Handler gefüllt. Am Ende der Verarbeitung wird die Konvertierung in XML gestartet. Die folgende Tabelle zeigt den beispielhaften Aufbau des Puffers für Knoten.

Tabelle 17 Auszug aus Puffertabelle für Knoten (Hauptspeicher)

Orgunit	Produnit	Level	Display
A	0815	1	X
A	1234	5	
B	0815	2	X
B	4711	6	
B	4718	9	
C	0911	5	X

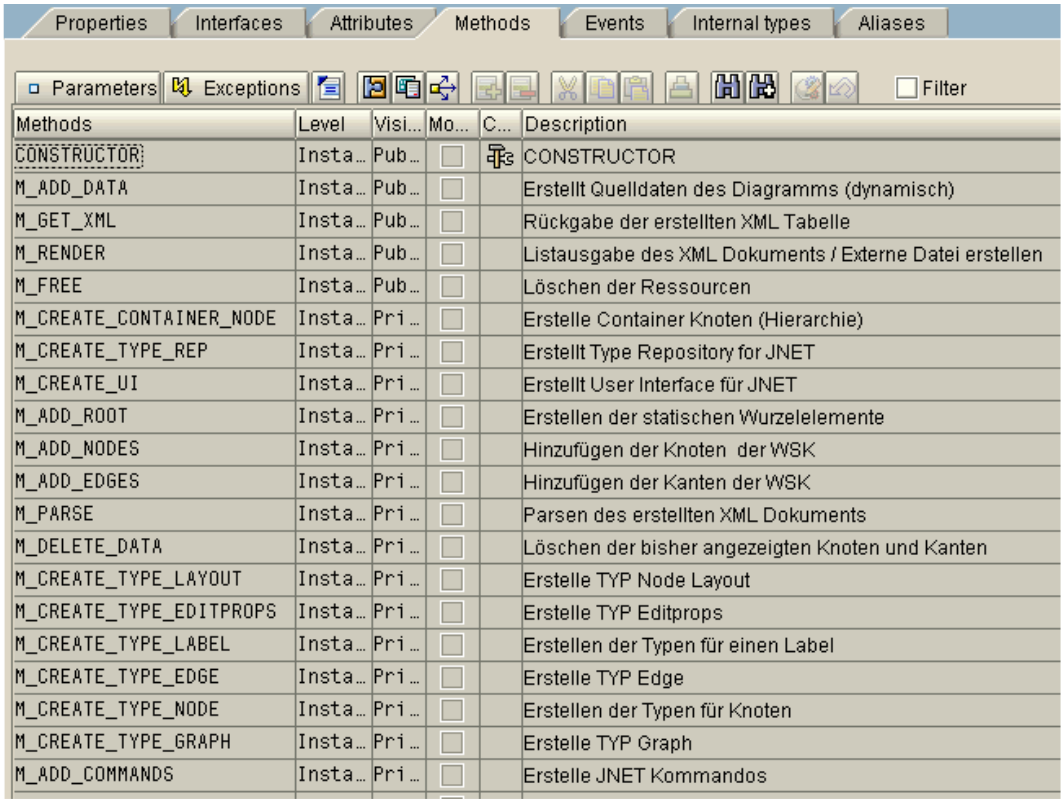
Die beiden Puffertabellen werden nicht persistent auf der Datenbank gespeichert, sondern für jeden Programmstart einmalig in den Hauptspeicher geladen.

6.4 XML Handler

Nachdem die Daten korrekt ermittelt wurden, musste eine Konvertierung in XML entwickelt werden. Diese Aufgabe wurde in einer eigenen Klasse gekapselt. Der folgende Abschnitt beschreibt den Ablauf und die Ergebnisse der Entwicklung des XML-Handlers.

6.4.1 Entwicklung der Komponente

Die Klasse XML Handler wurde ebenfalls im Class Builder als DDIC Objekt erstellt. Werden Änderungen an dieser Klasse vorgenommen, sollen diese in allen Programmen, die die Visualisierung verwenden, übernommen werden. Die Klasse wurde, wie alle anderen Bestandteile der Visualisierung, der entsprechenden Entwicklungsklasse für JNet hinzugefügt. Die folgende Abbildung zeigt einen Screenshot des Class Builders mit einem Teil der Klassendefinition des XML-Handlers (s. Abbildung 39).



Methods	Level	Visi...	Mo...	C...	Description
CONSTRUCTOR	Insta...	Pub...			CONSTRUCTOR
M_ADD_DATA	Insta...	Pub...			Erstellt Quelldaten des Diagramms (dynamisch)
M_GET_XML	Insta...	Pub...			Rückgabe der erstellten XML Tabelle
M_RENDER	Insta...	Pub...			Listausgabe des XML Dokuments / Externe Datei erstellen
M_FREE	Insta...	Pub...			Löschen der Ressourcen
M_CREATE_CONTAINER_NODE	Insta...	Pri...			Erstelle Container Knoten (Hierarchie)
M_CREATE_TYPE_REP	Insta...	Pri...			Erstellt Type Repository for JNET
M_CREATE_UI	Insta...	Pri...			Erstellt User Interface für JNET
M_ADD_ROOT	Insta...	Pri...			Erstellen der statischen Wurzelelemente
M_ADD_NODES	Insta...	Pri...			Hinzufügen der Knoten der WSK
M_ADD_EDGES	Insta...	Pri...			Hinzufügen der Kanten der WSK
M_PARSE	Insta...	Pri...			Parsen des erstellten XML Dokuments
M_DELETE_DATA	Insta...	Pri...			Löschen der bisher angezeigten Knoten und Kanten
M_CREATE_TYPE_LAYOUT	Insta...	Pri...			Erstelle TYP Node Layout
M_CREATE_TYPE_EDITPROPS	Insta...	Pri...			Erstelle TYP Editprops
M_CREATE_TYPE_LABEL	Insta...	Pri...			Erstellen der Typen für einen Label
M_CREATE_TYPE_EDGE	Insta...	Pri...			Erstelle TYP Edge
M_CREATE_TYPE_NODE	Insta...	Pri...			Erstellen der Typen für Knoten
M_CREATE_TYPE_GRAPH	Insta...	Pri...			Erstelle TYP Graph
M_ADD_COMMANDS	Insta...	Pri...			Erstelle JNET Kommandos

Abbildung 39 Klassendefinition XML Handler im SAP Class Builder

Zunächst wurde ein zentrales XML-Dokument definiert. Danach wurden die statischen Inhalte des XML-Dokuments erzeugt. Hierbei handelt es sich um das Type Repository und das User Interface. Der Inhalt dieser beiden Teile des XML muss nur einmal pro Laufzeit erzeugt werden. Da das Type Repository sehr umfangreich wurde, aber einer bestimmten Struktur folgte, konnte für die Erstellung jedes Typs eine eigene Methode verwendet werden. Dies ermöglichte eine einfache Pflege der Typen. Die Teilbäume TypeRepository und UserInterface wurden an den Beginn des zentralen XML-Dokuments gesetzt.

Einen wesentlichen Anteil des XML-Dokuments bildet jedoch der dynamische Inhalt. Dieser besteht aus den relevanten Knoten und Kanten. Die dynamischen Daten werden den statischen Komponenten angehängt und bei jeder Konvertierung neu erzeugt. Da einige Kanten bzw. Knoten besonders dargestellt werden sollen, mussten einige Zusatzprüfungen entwickelt werden. Um z.B. die Unterscheidung von Transfer und Verbrauch feststellen zu können, mussten die FI für den entsprechenden Knoten nachgelesen werden.

Um erste Tests des Datenproviders und des XML-Handlers zu ermöglichen, wurde zunächst eine Methode entwickelt, die das fertige XML-Dokument auf dem Server ablegte und zusätzlich auf dem Bildschirm ausgab. Das XML-Dokument wurde mit der standalone JNet-Applikation geöffnet und die Darstellung überprüft. Dies hatte den Vorteil, dass kleine Änderungen schnell geprüft und Strukturfehler im Dokument besser erkannt werden konnten. Diese Methode kann ebenfalls dazu verwendet werden, dem Benutzer das Speichern in eine externe XML-Datei zu erlauben.

Nachdem das XML-Dokument korrekt erstellt wurde, musste die Verbindung zwischen dem XML Handler und dem Visualisierungsfenster erfolgen. Hierfür wurde in der Klasse XML Handler eine GET-Methode definiert, die das XML-Dokument in Form einer internen Tabelle zurückliefert. Die Klasse CL_JNET holt sich über den Datenprovider die interne Tabelle mit dem XML ab.

6.4.2 Beschreibung des Ergebnisses

Das konkrete Ergebnis der Klasse XML Handler besteht aus dem fertigen XML-Dokument. Dieses setzt sich aus den drei wesentlichen Elementen Type Repository, User Interface und Graph(Daten) zusammen. Die folgende Abbildung (s. Abbildung 40) zeigt das DOM des erstellten XML-Dokuments.

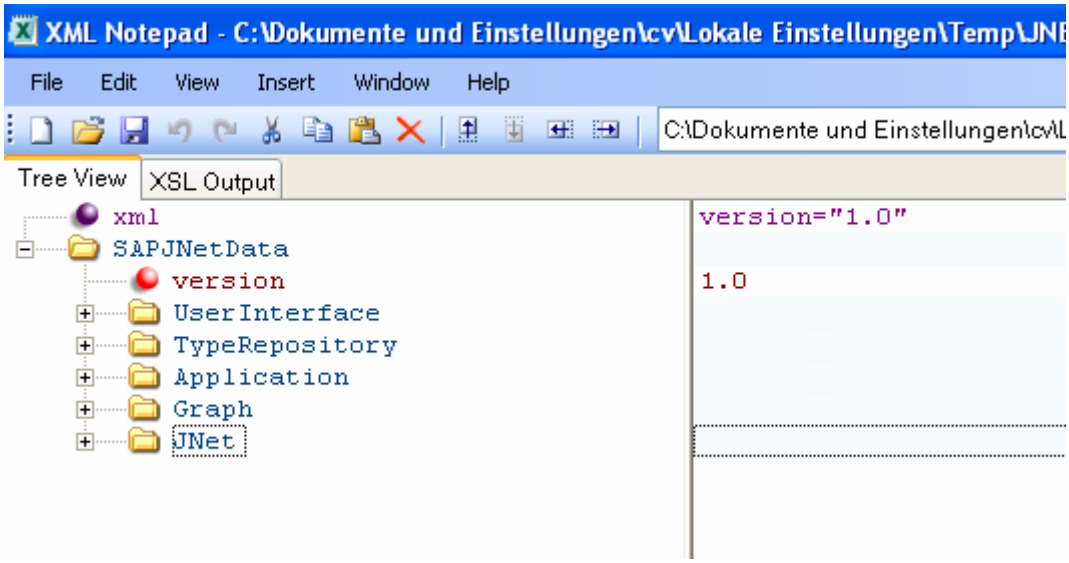


Abbildung 40 DOM Darstellung des erzeugten XML Codes

Neben den zuvor erwähnten Elementen finden sich in der Grafik das Wurzelelement SAPJNetData und die beiden weiteren Elemente Application und JNet. Das Wurzelelement sorgt für die Wohlgeformtheit des Dokuments, das Element Application beschreibt, welches Basismodell verwendet wird und das JNet Element enthält Kommandos, die beim ersten Anzeigen des Graphen ausgeführt werden sollen.

Das fertige XML-Dokument steht standardmäßig als DOM-Objekt zur Verfügung. Mit einigen Zusatzmethoden kann dieses DOM-Objekt in eine Datei gespeichert oder in eine interne Tabelle konvertiert werden.

6.5 Exemplarische Einbindung in GCP Anzeigewerkzeug

Eine der Anforderungen war es, die Visualisierung in mehrere Applikationen einbinden zu können. Dies ist auch möglich, da das JNet Navigationsfenster durch einen Include in jedes Programm eingebunden werden kann. Der entwickelte Prototyp sollte zunächst in die GCP CI-Anzeigefunktion eingebunden werden (s. Kapitel 3.2). Da das JNet-Navigationsfenster nicht von Beginn an angezeigt werden sollte, musste in der Basisapplikation zunächst ein entsprechender Absprung definiert werden. Zunächst wurde jedoch eine Kopie der originalen Anzeigefunktion erstellt, die als Testbasis dienen sollte.

In der Anzeigefunktion wurde ein Button vorgesehen, mit dem man die Anzeige der Wertschöpfungskette starten kann. Erst wenn dieser Button gedrückt wird, läuft die JNet Verarbeitung. Wird keine Visualisierung gewünscht, hat man somit nicht mit einer zusätzlichen Ressourcenbelastung zu rechnen.

In der PAI (Process After Input) Verarbeitung des Buttons der Visualisierung wurde ein weiterer Include eingebunden, der die Steuerung des JNet Objekts übernimmt. Hierbei wurde sichergestellt, dass das Navigationsfenster nur einmal pro Programmstart als Objekt erzeugt wird. Ändern sich die Anzeigedaten, wird das vorhandene Objekt mit den entsprechenden Methoden modifiziert. Neben dem Include zur Steuerung des Navigationsfensters, mussten die JNet Datendefinitionen in die Anzeigefunktion eingebunden werden. Die Datendefinitionen wurden ebenfalls in einem eigenen Include gekapselt, der in die globale Datendefinition, den sog. TOP Include, der Anzeigefunktion integriert wurde. Im Include für die Datendefinitionen sind weiterhin die Klassen CL_JNET und CL_EVH implementiert.

Schließlich wurde noch ein Include definiert, der alle Ressourcen der Visualisierung beim Beenden der Anzeigefunktion wieder freigibt. Die folgende Abbildung zeigt die Visualisierung der Wertschöpfungskette eingebunden in die Anzeigefunktion (s. Abbildung 41).

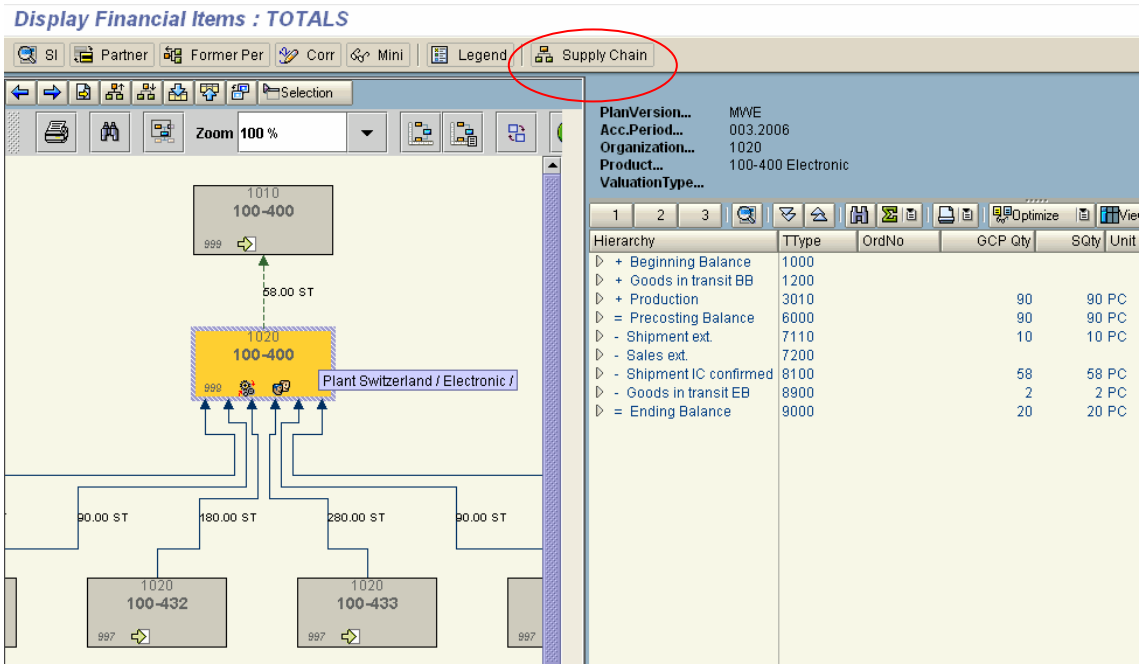


Abbildung 41 Screenshot JNet Visualisierung in Anzeigefunktion

Es wurden neben den eigentlichen Programmanpassungen auch einige Änderungen am Selektionsbildschirm der Applikation notwendig, da die Visualisierung mit JNet mit zusätzlichen Parametern versehen werden kann. Ein Beispiel hierfür ist die Anzahl der anzuzeigenden Levels.

6.6 Test der Entwicklung

Im Zuge der Entwicklung des JNet Visualisierungswerkzeuges kam es zu vielen Tests. Diese wurden nicht nur nach der Fertigstellung des Ganzen durchgeführt, sondern auch auf einzelne Komponenten bezogen. Der Vorteil hierbei war, dass die einzelnen Komponenten voneinander unabhängig testbar waren. So konnte das Zusammenspiel von Datenprovider und XML Handler sehr einfach durch die Analyse des resultierenden XML-Dokuments getestet werden. Da JNet nicht innerhalb eines SAP GUI laufen muss, sondern auch eine eigenständige Applikation mitbringt, konnte das erstellte XML-Dokument in diese Applikation geladen werden, um den Graph zu testen.

Die Anzeige innerhalb des SAP GUI, im HTML Control, wurde zunächst mit den statischen Beispielmodellen getestet. Später wurden diese durch Tests mit den dynamischen Daten ergänzt. Um die Applikation jedoch richtig testen zu können, mussten zunächst einige Testdaten erstellt werden. Sowohl im Entwicklungssystem, als auch in der Konsolidierung, standen bestimmte Testmodelle zur Verfügung. Diese Modelle bestanden aus angepassten Kundendaten. Da die Testdaten auf der Entwicklungsmaschine relativ alt waren, wurde jeder fertige Entwicklungsstand in das Konsolidierungssystem transportiert, welches über aktuellere Daten verfügt. Dort konnten somit erweiterte Prüfungen durchgeführt werden, deren Erkenntnisse wiederum in neue Entwicklungen und Korrekturen einfließen konnten.

Die bisher implementierten Funktionen wurden mit den entsprechenden Daten mehrfach gestestet. Bis auf kleinere Fehler, die schnell behoben werden konnten, läuft die Applikation stabil.

Im Hinblick auf die Performance konnte subjektiv keine Verschlechterung festgestellt werden. Um dies objektiv zu belegen, wurden einige Laufzeitmessungen in der ABAP Runtime Analysis durchgeführt. Die folgende Abbildung(s. Abbildung 42) zeigt das Ergebnis der Laufzeitanalyse des JNet Testprogramms.

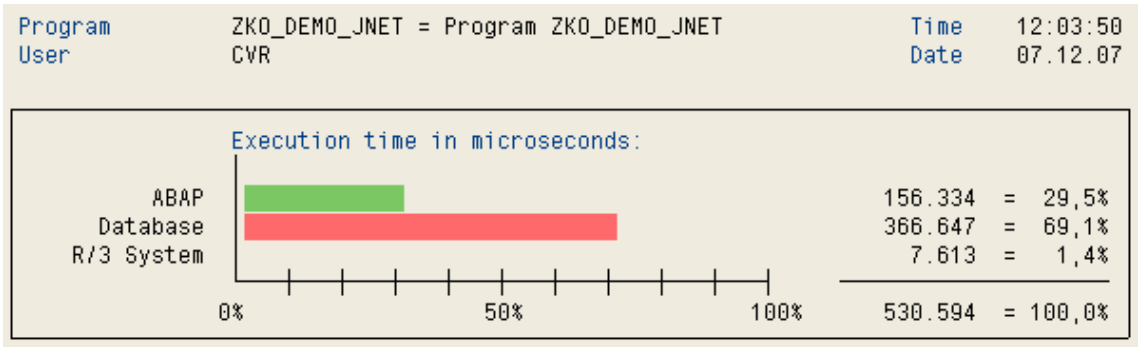


Abbildung 42 Zusammenfassung Laufzeitanalyse JNet Fenster

Als Ergebnis liefert die Laufzeitanalyse ein Balkendiagramm, welches die Gesamtlaufzeit anteilig der Programmart unterscheidet. Die Art besteht hierbei aus ABAP (Laufzeit im eigentlichen Programmcode), Database (Gesamtzeit für Datenbankzugriffe) und R/3 System (Anteil Systemverarbeitung). Der Systemanteil kann jedoch in diesem Programm vernachlässigt werden. Den Großteil der Laufzeit macht die Datenbankselektion aus, da die Knoten und Kanten von der Datenbank selektiert werden müssen. Die Zusatzinformationen, die in Form von FIs für manche Knoten nachgelesen werden, vergrößern ebenso die Datenbanklaufzeit. Der ABAP

Anteil der Gesamtlaufzeit ist ebenso sehr gering. Hier wird der Großteil des Aufwandes durch das dynamische Erstellen des XML-Dokuments erzeugt. Die Messung wurde für das Navigationsfenster in einer simplen Rahmenanwendung durchgeführt. Auf der linken Seite der Abbildung sind die jeweiligen Mikrosekunden dargestellt. Die angefallene Summe von 530.594 Mikrosekunden Laufzeit, dies entspricht ca. einer halben Sekunde, befindet sich in einem guten Rahmen.

In einem weiteren Test wurde die Gesamtlaufzeit der neuen Anzeigefunktion mit der integrierten Visualisierung getestet. Hierbei ist zu beachten, dass die CI Anzeigefunktion selbst eine sehr komplexe Applikation ist. Die folgende Abbildung (s. Abbildung 43) zeigt das Ergebnis dieses Tests.

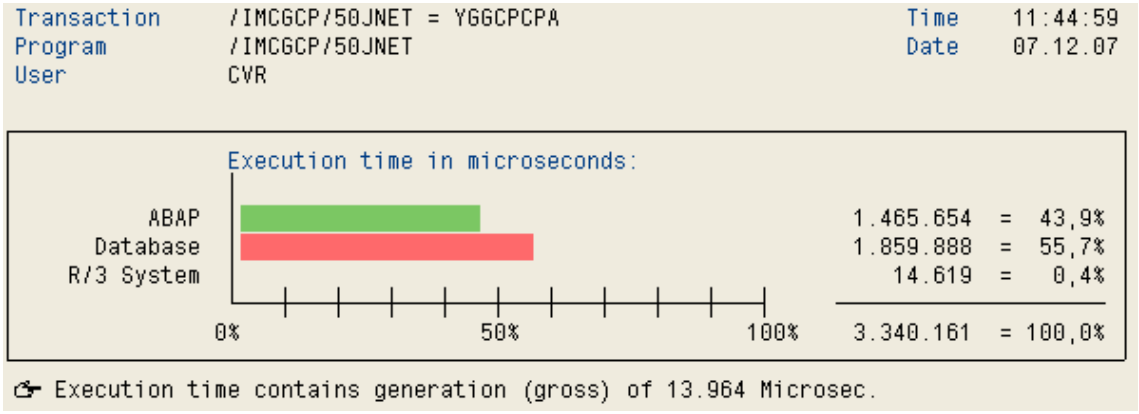


Abbildung 43 Laufzeittest Anzeigefunktion mit Visualisierung

Die Gesamtlaufzeit der Anwendung ist im Falle der Anzeigefunktion mit Visualisierung deutlich höher als zuvor. Dies ist allerdings zu erwarten, da die Anzeigefunktion selbst bereits knapp zwei Sekunden Laufzeit für sich beansprucht. In diesem Fall belief sich die Laufzeit auf insgesamt fast drei Sekunden. Datenbank und ABAP Anteil sind nahezu gleich, wodurch man auf eine hohe ABAP Laufzeit in der Anzeigefunktion schließen kann.

Im Anhang dieser Arbeit findet sich ein detailliertes Performance und Testprotokoll mit mehreren Testfällen. Hierbei werden zusätzlich die einzelnen Optionen für die Anzeige mit JNET unterschieden. Weiterhin wurde die Laufzeit einer Navigation durch mehrere Stufen getestet.

7 Resultate und Zusammenfassung

Dieses Kapitel bildet den Abschluss der Arbeit. Hier sollen die Vorgehensweise und die Ergebnisse der Entwicklung kritisch betrachtet werden. Weiterhin soll vorgestellt werden, welche Methoden besonders erfolgreich waren und was bei kommenden Projekten besser gemacht werden könnte.

7.1 Zusammenfassung der Lösungserarbeitung

Zu Beginn des vorliegenden Dokuments wurde zunächst eine Einführung in die verwendeten Techniken bzw. den betriebswirtschaftlichen Hintergrund gegeben. Hierbei wurde besonders auf die GCP eingegangen und herausgearbeitet, an welchen stellen eine Visualisierung sinnvoll ist.

Anforderungen und Spezifikation
Alle Ergebnisse und Erkenntnisse der im Projektvorlauf durchgeführten Meetings flossen in eine zentrale Software-Spezifikation zur Visualisierung einer globalen Wertschöpfungskette ein. Hierbei wurde zunächst eine Definition der primären und sekundären Use-Cases vorgenommen (s. Abbildung 10, S. 24). Diese dienten der Definition der funktionalen Anforderungen. Da sich die Use-Cases im Detail in zahlreiche funktionale Anforderungen trennen, wurde eine besondere Darstellung gewählt, um diese einzelnen Features in Funktionsgruppen zu gliedern (s. Abbildung 11 S. 25). Das vorliegende Dokument erfüllt das Ziel als zentrales Spezifikationsdokument zu dienen, welches die einzelnen Anforderungen zusammenführt.

Technische Evaluation
Die Visualisierung einer globalen Wertschöpfungskette kann mit unterschiedlichen Techniken ermöglicht werden. Das Ergebnis der zu Beginn des Projekts durchgeführten Technologieevaluation war das Werkzeug JNet, welches ebenso von der SAP als Anwendung der Zukunft empfohlen wurde. JNet hat hierbei den Vorteil eigenständig betrieben werden zu können und sehr flexibel für Erweiterungen zu sein.

Design der Software
Einen weiteren Bestandteil dieses Dokuments bildet das Design der Software zur Visualisierung globaler Wertschöpfungsketten. Im Design wurden zunächst die drei Hauptkomponenten der Applikation in einem Komponentendiagramm vorgestellt und deren Schnittstellen angedeutet. In einem nächsten Schritt wurde eine Übersicht über die einzelnen Klassen gegeben, aus denen die Komponenten bestehen. Die im Komponentendiagramm (s. Abbildung 28, S. 51) angedeuteten Schnittstellen wurden in einem speziellen Kapitel (vgl. Kapitel 5.4) näher vorgestellt und deren Parameter erläutert. In diesem Abschnitt wurde ebenso eine detaillierte Beschreibung der Klassen vorgenommen. Weiterhin wurden Sequenzdiagramme erstellt, die einen Einblick in das dynamische Verhalten der Applikation gewähren sollen. Zum Abschluss des Design-Kapitels wurden einige Designalternativen vorgestellt. Das Designkapitel konnte somit das Ziel erreichen, eine solide Basis zur Entwicklung des Prototyps zu bilden und gleichzeitig eine ausführliche technische Dokumentation darzustellen.

Realisierung
Die Realisierung zeigt die Vorgehensweise bei der Entwicklung des Prototyps. Hierbei wurde zunächst ein kleines Testprogramm erstellt, welches JNet auf die gestellten

Anforderungen hin testen sollte. Nachdem dieser Test erfolgreich war, wurde dieses Testprogramm zum jetzigen Prototyp erweitert. Da die Visualisierung in mehreren Applikationen verfügbar sein sollte, wurde die konkrete Implementierung der Visualisierung in die GCP Anzeigefunktion vorgestellt. Den Abschluss der Realisierung bildeten die Funktions- und Performancetests (vgl. Kapitel 6.6). Hierbei konnte festgestellt werden, dass alle angestrebten Funktionen in der Testumgebung funktionierten. Die Performance der Applikation liegt in einem guten Rahmen, da die Laufzeit der Anzeigefunktion nur unwesentlich gestiegen ist.

Das Ergebnis der Entwicklungen ist ein Prototyp, der in mehrere Applikationen eingebunden werden kann. Die primären Use-Cases sind voll unterstützt, da der Benutzer durch die Wertschöpfungskette navigieren kann und hierbei auch nur relevante Ausschnitte betrachtet. Weiterhin wird eine Interaktion mit der Anzeige ermöglicht, da das Modell dynamisch erweitert werden kann bzw. ein Zoom Details ein- und ausblendet. Die Darstellung der einzelnen Bestandteile der Wertschöpfungskette ermöglicht eine vereinfachte Auswertung der Wertschöpfungsstruktur. Außerdem können Fehler, die GCP beim Aufbau der Wertschöpfungskette erzeugt hat, bzw. Dateninkonsistenzen schneller nachvollzogen werden. Somit kann der Prototyp die in Kapitel 3 vorgestellten Anforderungen weitestgehend erfüllen. Aufgrund der verwendeten Technik waren manche Anforderungen jedoch nicht 1:1 realisierbar.

Neben den eigentlichen Anforderungen wurden erste Funktionalitäten aus dem Ausblick realisiert. Somit wurde bereits ein erstes Customizing implementiert, welches dem Benutzer erlaubt, eigene Parameter für Farben, Größe und Zoomgewicht der einzelnen Komponenten des Graphs zu setzen.

An dieser Stelle wurde die Anzeige von Rekursionen, die (zum Teil) außerhalb des Kontexts liegen, im Prototyp noch nicht implementiert. Um alle Zyklen in einem solchen Graph erkennen zu können, bedarf es einem komplexen Algorithmus, dessen Laufzeit, insbesondere bei größeren Wertschöpfungsketten, nicht für Echtzeit-Anwendungen geeignet ist.

7.2 Kritische Betrachtung

In diesem Kapitel soll der Verlauf des Projekts kritisch betrachtet werden. Hierbei wird auf die Spezialitäten der erstellten Lösung eingegangen und die aufgetretenen Probleme erläutert. Weiterhin wird dargestellt, welche Maßnahmen und Vorgehensweisen erfolgreich waren und was in folgenden Projekten verbessert werden könnte.

7.2.1 Komponente

In diesem Abschnitt wird das Ergebnis der Arbeit näher betrachtet. Das Ergebnis ist in diesem Zusammenhang der erstellte Prototyp. Vor dem Start der Entwicklung stellte sich die Frage, wie JNet in einen SAP GUI integriert werden kann. Hier lässt sich die erste Besonderheit der Applikation identifizieren. Die Kommunikation zwischen JNet und dem SAP GUI wurde durch ein spezielles Vorgehen in der Ereignisverarbeitung ermöglicht. Eine weitere Besonderheit bildet die interne Datenhaltung der Wertschöpfungsausschnitte. Es wurde eine Puffertabelle vorgesehen, die die Daten bei einer Navigation schneller zur Verfügung stellen soll. Diese Puffertabelle steigert die Performance bei der Navigation in der Wertschöpfungskette sehr deutlich. Weiterhin werden die bisher angezeigten Modelle gespeichert, um ein Vor- bzw.

Rückwärtsspringen, ähnlich wie in einem Browser, zu ermöglichen. Die gewählte Programmiertechnik unterstützte eine effiziente Entwicklung der Funktionen im Sinne der Wiederverwendung.

Besonderheiten der Lösung
Die Visualisierung der Wertschöpfungskette soll in mehreren Applikationen verfügbar sein. Ein spezieller Mechanismus ermöglicht die individuelle Definition von Diensten für jede Rahmenapplikation. Außerdem wurde die Visualisierung in ein eigenes Entwicklungspaket gekapselt, was eine problemlose Wiederverwendung in beliebigen Rahmenapplikationen ermöglicht. Probleme beim Transport in Kundensysteme können somit ebenso verhindert werden.

JNet hat durch seine Unabhängigkeit vom SAP GUI einen weiteren Vorteil. Fertige Wertschöpfungsgraphen, die als XML-Dokument vorliegen, können in einer eigenen unabhängigen Applikation angezeigt werden. Dies erleichtert die Kollaboration in der Wertschöpfungskette. Lieferanten und Kunden können, falls gewünscht, einen gegenseitigen Einblick in die Wertschöpfungsstruktur erhalten.

Ein weiteres Feature der Applikation besteht aus dem zusätzlichen Anzeigen der Stammdaten aller Knoten. Hierbei werden Informationen über Werk und Material aus den GCP-Quelltabellen gelesen und zur Anzeige mit aufbereitet. Des Weiteren werden dem Knoten Symbole hinzugefügt, die Rückschlüsse auf die Beschaffungsalternative zulassen. Hierbei werden Produktion und Einkauf unterschieden. Weitere Icons zeigen an, ob das Produkt direkt zum Kunden verkauft wird und ob es sich in einer direkten Rekursion befindet. Alle diese Funktionen lassen sich im eigenen Customizing ein- bzw. ausschalten.

Verbesserungspotential der Lösung
Insgesamt bietet der Prototyp die angeforderten Funktionalitäten an. Es bestehen jedoch einige Möglichkeiten zur Erweiterung und Verbesserung dieser Applikation. Diese Erweiterungen werden in folgenden Projekten geprüft und ausgearbeitet.

Die erste Problematik ergibt sich aus der Technik JNet selbst. Der Kunde muss derzeit auf seinem lokalen Rechner einige Voraussetzungen zum Betrieb von JNet treffen (vgl. Kapitel 5.1).

Einen weiteren Punkt zur Verbesserung beinhaltet die Unterscheidung der Beschaffungsalternativen. Die unterschiedlichen Alternativen werden zwar durch Symbole visualisiert, die Kantenmengen werden hierbei jedoch nicht anteilig unterschieden. Sind beispielsweise mehrere Alternativen vorhanden, trägt die Kantenbeschriftung nur die Gesamtmenge.

JNet ist in der Lage sehr viele Knoten und Kanten in einem Graphen darzustellen. Da die Wertschöpfungskette sowohl topdown, als auch bottomup durchlaufen wird, kann es, insbesondere bei der Visualisierung auf Rohstoffebene, zur Anzeige vieler Knoten kommen. Diese lassen den Graphen schnell unübersichtlich werden. An dieser Stelle sollte ein Mechanismus eingebaut werden, der die Anzeige auf eine bestimmte Gesamtzahl an Knoten eingrenzt. Die aktuelle Anwendung zur Visualisierung ermöglicht zwar die Anzeige der kompletten Wertschöpfungskette, es sollte jedoch bei

komplexeren Modellen auf die Ausschnittsanzeige zurückgegriffen werden, da das Aufbereiten aller Knoten und Kanten dann sehr langsam werden kann.

Ein weiterer verbesserungswürdiger Punkt besteht in der Darstellung der Vorgänger- bzw. Nachfolgerknoten. Es wurde angestrebt, die noch vorhandenen Nachfolger bzw. Vorgänger zu zählen. Aus Gründen der Performance wird jedoch immer nur der aktuelle Ausschnitt analysiert. Dies macht das Zählen aller Nachfolger im Modell nicht mehr möglich.

7.2.2 Vorgehensweise

Besonderheiten im Vorgehen
Ein großer Vorteil in der Vorgehensweise dieses Projekts lag in der Recherche vorab. Hierbei wurden zunächst Technologien gesucht, die für eine solche Visualisierung in Frage kamen. Es gelang schnell, auch in Absprache und Zusammenarbeit mit der SAP, einige Techniken ausfindig zu machen, wobei JNet schon vorab als Empfehlung ausgesprochen wurde.

Nachdem JNet als Technik feststand, war es sehr gut, zuerst ein kleines Testprogramm zu erstellen, um die Funktionalität und Arbeitsweise innerhalb des SAP GUI zu testen.

Weiterhin war es sehr hilfreich, den Prototypen in einzelnen Stufen zu entwickeln. Somit konnten die ersten Tests, abgegrenzt für jede einzelne Stufe, durchgeführt werden. Nachdem eine Stufe abgeschlossen und gestestet wurde, konnte mit der nächsten Stufe begonnen werden.

Die Unterstützung der SAP AG für dieses Projekt war ein weiterer wichtiger Faktor während der Projektdurchführung. Da JNet ein sehr „junges Werkzeug“ ist, wurde es noch nicht sehr umfangreich getestet. Einige JNet-Funktionen zeigten während der Realisierung ein unerklärbares Verhalten, was auf interne Programmfehler zurückzuführen war, die an die SAP gemeldet und direkt vom zuständigen Entwickler korrigiert wurden. Manche Funktionalitäten wurden auf Basis der Anforderungen dieses Projekts in das JNet-Framework als Standardfunktion übernommen. Hier ist besonders die Zoomgewicht-Funktion zu nennen, die es ermöglicht, die Komponenten eines Graphs ab einer bestimmten Zoomstufe ein bzw. auszublenden. Somit entstand eine Win-Win-Situation durch die enge Zusammenarbeit in diesem Projekt.

Verbesserungspotential in der Vorgehensweise
Trotz des insgesamt guten Ablaufs des Projekts, gab es einige Punkte innerhalb der Vorgehensweise, die zu verbessern wären. Durch erste Demos der Visualisierung und dem Dialog mit einigen Pilotbenutzern, entstanden weitere Ideen, die Einfluss in die Entwicklung nahmen, aber nicht Bestandteile der eigentlichen Anforderungen waren. Durch die Berücksichtigung dieser Ideen verlängerte sich die Entwicklungszeit. Es gab somit weitere Funktionen, die zu testen waren. An dieser Stelle wäre es besser gewesen, gezielt auf die Anforderungen zu verweisen und die neuen Ideen in ein weiteres Konzept für den Nachlauf aufzunehmen.

Innerhalb der Realisierung wurde der Fokus zu lange auf einzelne Programmteile gelegt. Somit dauerte es zu lange, bis eine Teilapplikation entstehen konnte, die bereits eine Idee der fertigen Lösung darstellen sollte. Als eine solche Teilapplikation für eine

Kundenpräsentation gefordert wurde, kam es somit zu unnötigem Termindruck. Nachdem einige Probleme gelöst wurden, stand jedoch schnell eine solche Teilapplikation zur Verfügung und es konnten erste Demonstrationen vorgenommen werden.

Da der Prototyp teilweise auf das zuvor erstellte Testprogramm aufbaut, kam es zu Abweichungen in der Klassendefinition. Hierbei unterscheiden sich zum einen die Klassennamen, aber auch die Namen von Methoden und Attributen. Die neuen Klassen sind in einem zentralen Repository verfügbar, während die vorhandenen Klassen inline im Programmcode definiert wurden.

Lessons learned
Am Ende dieses Abschnitts soll auf die „lessons learned“ im Projekt eingegangen werden. Hierbei sollen die Eindrücke und Erfahrungen beschrieben werden, die mitgenommen werden konnten.

Zunächst ist es sehr wichtig, eine solide Basis für ein Softwareprojekt zu bilden. Diese Basis besteht aus einem möglichst vollständigen Spezifikationsdokument, welches die Anforderungen an die Software konkret abbilden kann.

Das Design bildet das Standbein für einen erfolgreichen Prototypen bzw. eine erfolgreiche Software. Es ist also nicht sehr ratsam, mitten in der Realisierung neue Anforderungen mit aufzunehmen, da diese zu einem inkonsistenten Zustand in der Vorgehensweise führen. Sind, wie in diesem Fall, Folgeprojekte geplant, ist es unerlässlich, eine passende Dokumentation zur Verfügung zu stellen, die insbesondere auf die vorhandenen Schnittstellen und Erweiterungspunkte hinweist.

Es ist sehr wichtig, im Vorlauf von Projekten, Machbarkeitsanalysen und Tests durchzuführen, um so früh wie möglich zu erkennen, ob die gewählte Vorgehensweise zum gewünschten Erfolg führen kann. Während der gesamten Laufzeit sollte strukturiert vorgegangen werden, um den Überblick über den aktuellen Status zu behalten. Wichtige Schritte und Projekt-Milestones sollten immer im Auge behalten werden. Bei einer Überschreitung sind die Gründe zu analysieren. Weiterhin sollte man anstreben, den beteiligten Personen des Projekts schnellstmöglich eine Lösung zu erarbeiten, die eine Vorstellung über das fertige Ergebnis liefert. Somit kann man die Akzeptanz und das Interesse am Projekt deutlich verstärken.

Innerhalb des Projekts gilt es, feste Grenzen bzw. Endpunkte zu definieren. Werden vermehrt neue Punkte in die „ToDo“-Liste aufgenommen, schiebt sich das Projektende auf einen immer späteren Zeitpunkt hinaus. Durch solch unerwartete Verlängerungen kann letztendlich das gesamte Projekt gefährdet werden. In diesem Projekt wurde, nach der Ergänzung einiger sinnvoller Funktionen, das Ende des Projekts in Form einer Abschlussbesprechung festgelegt.

7.3 Erweiterungspotential

JNet bietet das Potential, einige GCP Funktionen künftig ganz grafisch zu steuern. Dabei würde JNet nicht mehr nur die Daten aus GCP Applikationen anzeigen, sondern diese auch modifizieren und an die entsprechende Applikation zurückliefern. Diese

Erweiterungen teilen sich in technisch-funktionale und qualitative Erweiterungen auf. Die folgende Grafik stellt die entsprechenden Erweiterungen gruppiert dar:

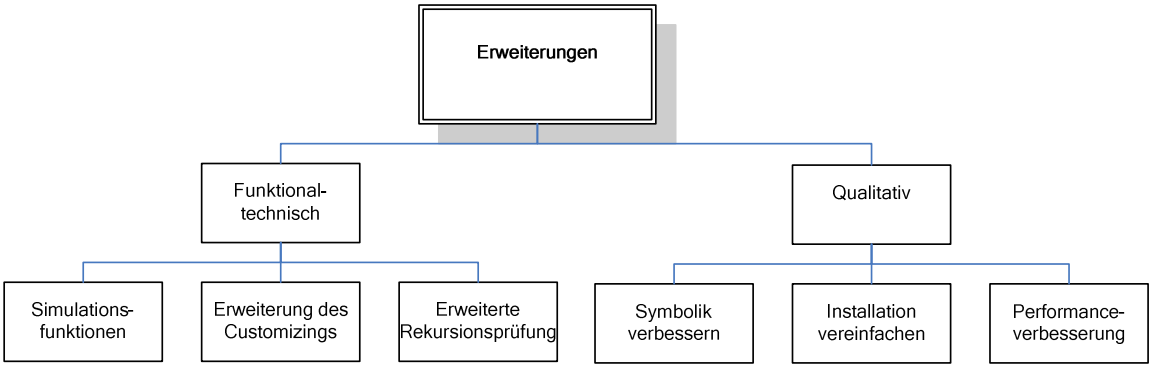


Abbildung 44 Ausblick und Erweiterungspotential

Funktional-technische Erweiterungen

In diesem Abschnitt werden die möglichen Features beschrieben, die den Funktionsumfang der Applikation erweitern können. Hierbei sind besonders die Simulationsfunktionen hervorzuheben, da diese die sinnvollste Ergänzung darstellen.

Die folgende Tabelle stellt die einzelnen funktionalen Erweiterungen vor:

Tabelle 18 Funktionale Erweiterungen der Visualisierung

Funktion	Beschreibung
Simulationsfunktionen	Die Simulationsfunktionen sollen es erlauben, aus JNet sämtliche Modifikationen an GCP-Daten vorzunehmen. *Mengensimulation:* Änderung von Mengen in Kanten und den einzelnen Knoten (Bsp.: Änderung einer Verbrauchsmenge von A nach B) *Preisänderung:* Änderungen von Einkaufs- bzw. Verkaufspreisen, Hinzufügen von Produktionskosten bzw. Währungsschwankungen (Bsp.: Erhöhen des Einkaufspreises eines Rohstoffes A) *Vorgangsänderung:* Änderungen in der Wertschöpfungsstruktur. Knoten bzw. Kanten fallen aus dem Modell heraus oder werden hinzugefügt (Bsp.: Ein Produkt A wird nicht mehr produziert sondern fremd beschafft). Nach Datenanpassungen aus JNet müssen ggf. einige GCP-Applikationen neu laufen
Erweiterte Customizing Einstellungen	- Erstellung eigener Layoutobjekte ermöglichen - Dynamische Anpassung von Knoten zur Laufzeit (Bsp.: Erscheinung eines Werkes abhängig von Parametern ändern: Werk A = blau und oval)

Funktion	Beschreibung
	- Pflegedialog für individuelles Customizing (View)
Erweiterte Rekursionsprüfung	Bisher können Rekursionen nur erkannt werden, wenn diese komplett innerhalb des betrachteten Ausschnitts liegen - Algorithmus implementieren, der Rekursionen in der ganzen Wertschöpfungskette feststellt - Speichern der betroffenen Sätze in einer Tabelle - Über Customizing betroffene Sätze markieren

Qualitative Erweiterungen an der Visualisierung
Die qualitativen Erweiterungen sollen den Umgang und die Installation dieser Anwendung erleichtern. Die folgende Tabelle beschreibt die in Abbildung 44 dargestellten Erweiterungen.

Tabelle 19 Qualitative Erweiterungen der Visualisierung

Funktion	Beschreibung
Erweiterung der Symbole innerhalb der Knoten	Innerhalb der Knoten werden im Moment einige Symbole angezeigt, die z.B. die Beschaffungsalternative darstellen - Erweiterung der Symbole (Preise, Bestand) - Kommentare bzw. Hilfe auf Symbolen definieren - Übersichtlichere Symbole verwenden
Vereinfachte Installation	- Automatische Installation der benötigten Daten auf dem lokalen Rechner bzw. Vermeiden einer lokalen Installation (zentrales Repository) - Fixe Pfadnamen verhindern - Reduzieren der benötigten Daten auf dem Zielrechner - Dokumentation für externes Programm zur Anzeige der Wertschöpfungskette (JNet standalone)
Performance	- Optimierung der Performance bei der Anzeige großer Wertschöpfungsketten (Alle Knoten) - Verbessertes Verhalten der Visualisierung beim ersten Programmstart (Initialisierung der JAVA-Umgebung)

Neben diesem Erweiterungspotential werden, falls sich der Prototyp bei Kunden bewährt, noch einige weitere Wünsche und Anforderungen folgen. JNet sollte aber den meisten davon gewachsen sein, da es sehr flexibel und erweiterungsfähig ist. Es steht eine Vielzahl von Schnittstellen zur Verfügung, die darauf warten, genutzt zu werden.

Literaturverzeichnis

Busch, Axel: Integriertes Supply Chain Management, 2. Auflage, Wiesbaden 2004.

Buschmann Frank et al.: Pattern-orientierte Softwarearchitektur, München 1998.

Chamoni, Peter; Gluchowski, Peter; Hahne Michael: Business Information Warehouse, Berlin usw. 2005.

Computer Data Institute (Hrsg.): SAP R/3 Einführung, München 2001.

Eckstein, Robert; Casabianca, Michel: XML kurz & gut, 2. Auflage, Köln 2002.

Flanagan, David: JavaScript – Das umfassende Referenzwerk, 5. Auflage, Köln 2006.

Gadatsch, Andreas: Grundkurs SAP R/3, Wiesbaden 2006.

Kuhn, Axel; Hellingrath Bernd: Supply Chain Management: Optimierte Zusammenarbeit in der Wertschöpfungskette, Berlin usw. 2002.

Musciano, Chuck; Kennedy Bill: HTML & XHTML - Das umfassende Referenzwerk, 5. Auflage, Köln 2003.

SAP AG: SAP Graphics (BC-FES-GRA), Walldorf 2001. (Auf CD enthalten)

Scholz-Reiter, Bernd; Jakobza Jens: Supply Chain Management. In: HMD – Praxis der Wirtschaftsinformatik, Nr. 207 vom Juni 1999.

Tempelmaier, Horst: Material-Logistik, 6. Auflage, Berlin usw. 2003.

Turau, Volker; Vogel, Friedrich: Algorithmische Graphentheorie, 2. Auflage, München, Wien 2004.

Ullenbom, Christian: Java ist auch eine Insel, 7. Auflage, Bonn 2007.

Van der Vlist, Eric: XML Schema, Cambridge usw. 2002.

Wurm, Fritz: IT-gestützte Konzernergebnisplanung und –steuerung. In: Controlling Nr. 6 vom Juni 2005. (Auf CD enthalten)

Interne Vorträge / Präsentationen:

Interner Vortrag GCP / Supply Chain Management (s. Ordner Quellen auf beigefügter CD)

Internetquellen:

Wikipedia: Wasserfallmodell:
http://upload.wikimedia.org/wikipedia/commons/e/e5/Wasserfallmodell.svg eingesehen am 09.11.2007.

SAP Intranet (s. Quelle im pdf Format auf beiliegender CD).

SAP Wiki Suchbegriff BADI:
https://www.sdn.sap.com/irj/sdn/wiki?path=/display/SRM/BAdI+-+general+information& eingesehen am 27.12.2007.

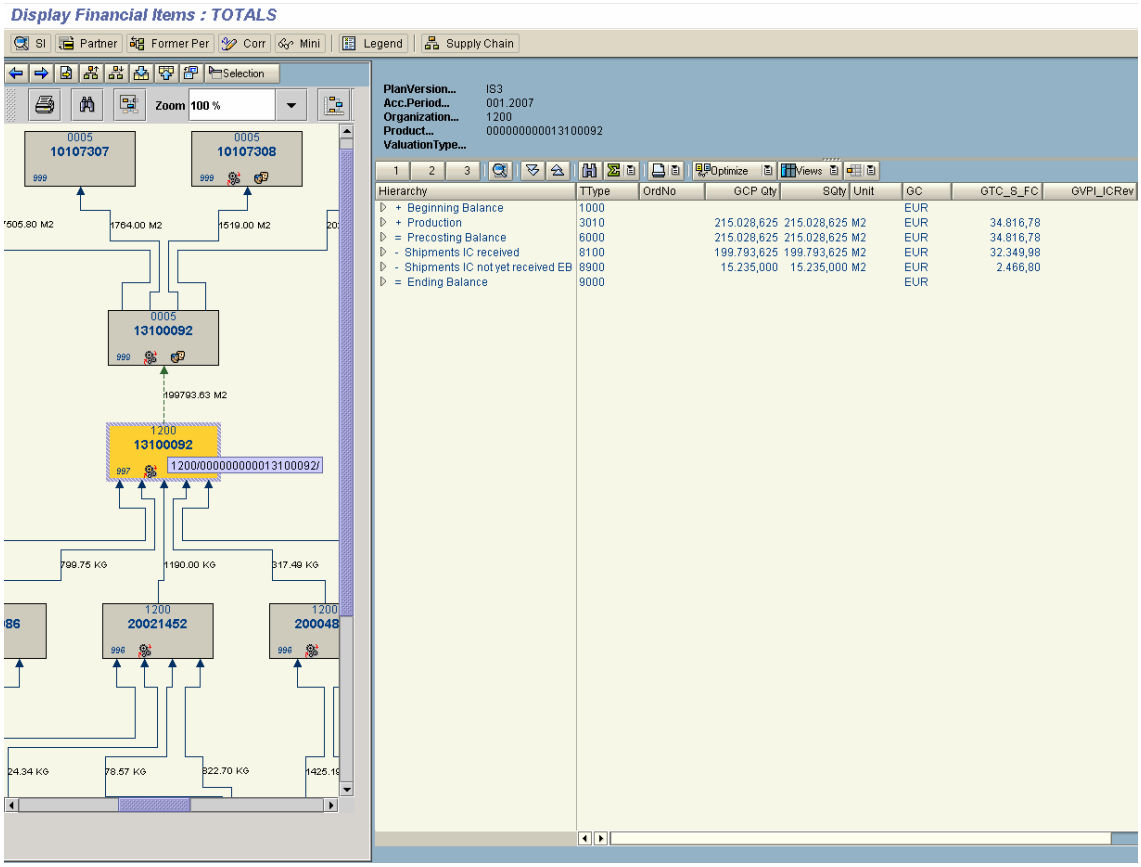
JNET / JGANTT Entwickler Dokumentation:
https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/lw/uuid/f010ec31-9658-2910-3c83-c6e62904eceb eingesehen am 19.09.2007.

HTML 4.01 Spezifikation des W3C:
http://www.w3.org/TR/html401/ eingesehen am 14.11.2007.

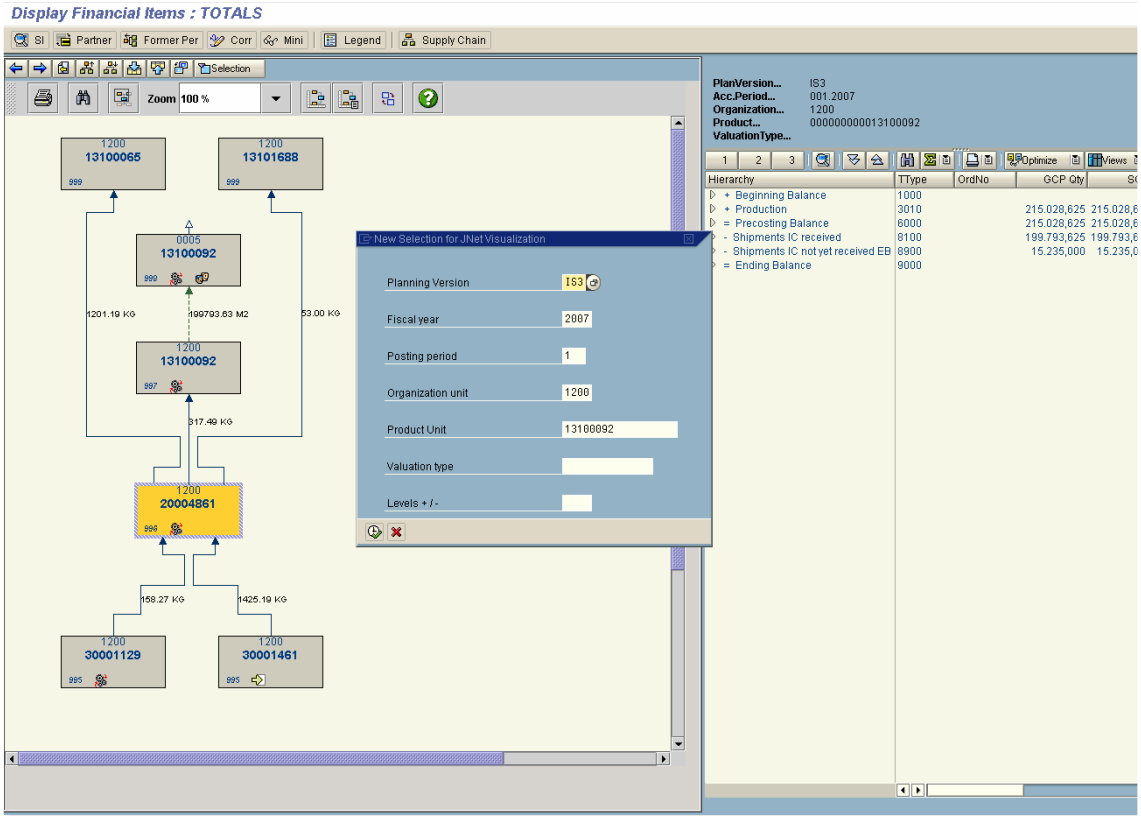
Bug Report von SUN, Problem JAVA Plugin im Internet Explorer:
http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4525262 eingesehen am 20.11.2007.

Anhang A

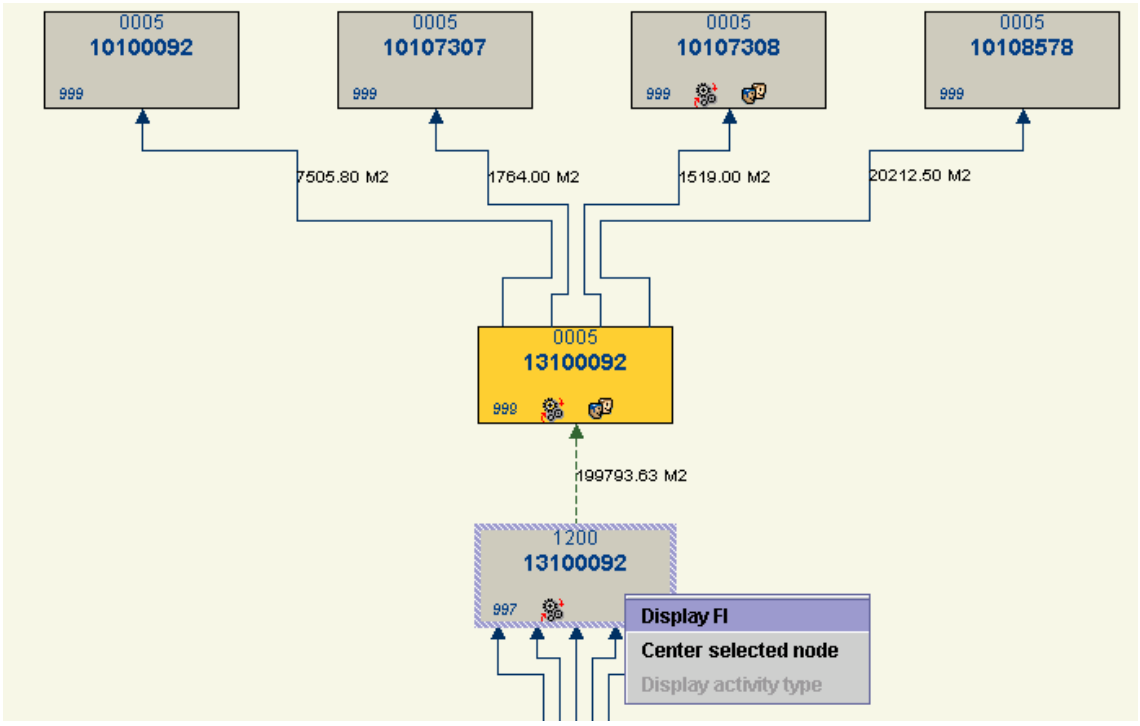
Screenshots des Ergebnisses:



Anzeigefunktion mit aktivierter Navigation



Anzeigefunktion mit ausgeklapptem Navigationsteil und Selektionsbildschirm zum Wechseln der Selektion.



Navigationskomponente mit Kontextmenü für Knoten.

Anhang B

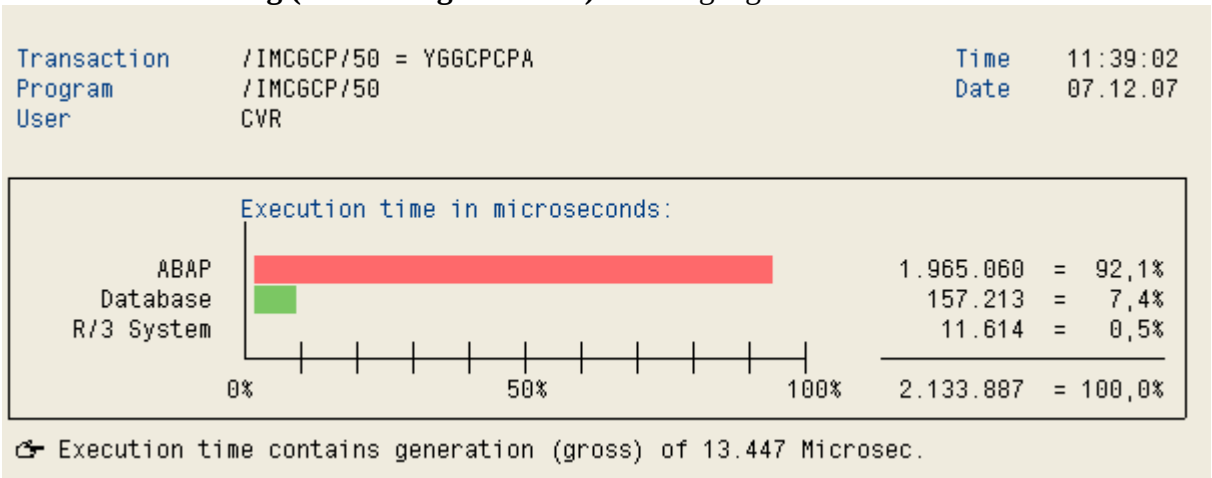
Testprotokoll der Visualisierung

Testprotokoll Designalternativen

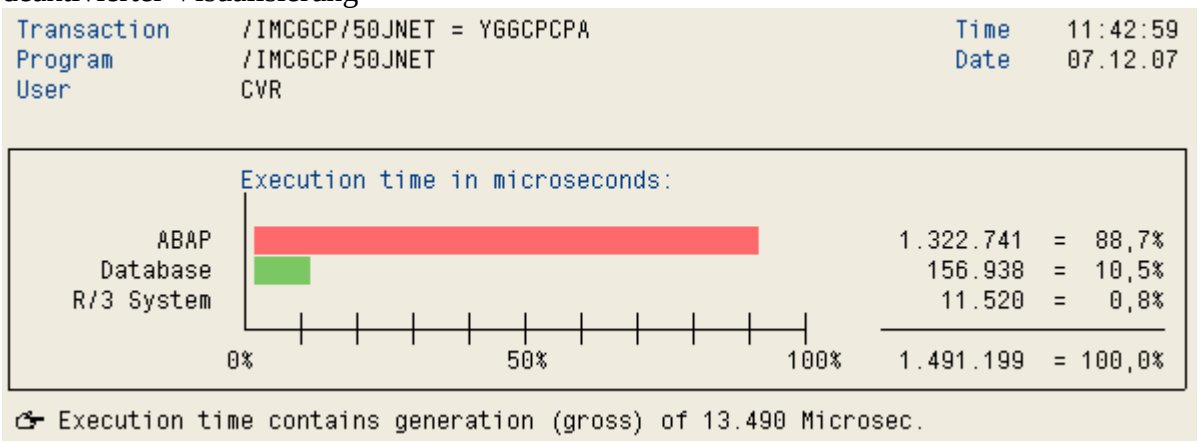
Anhang B Testprotokoll Visualisierung

Die Gesamtlaufzeit wird Mikrosekunden angegeben und wird in den Abbildungen rechts unten, unter dem Strich, abgebildet.

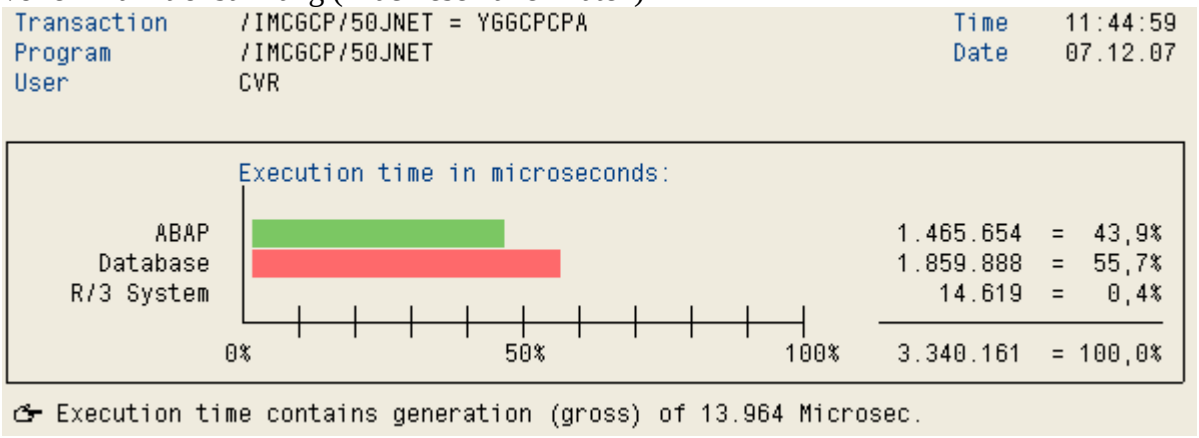
Ohne Visualisierung (alte Anzeigefunktion) => Ausgangssituation



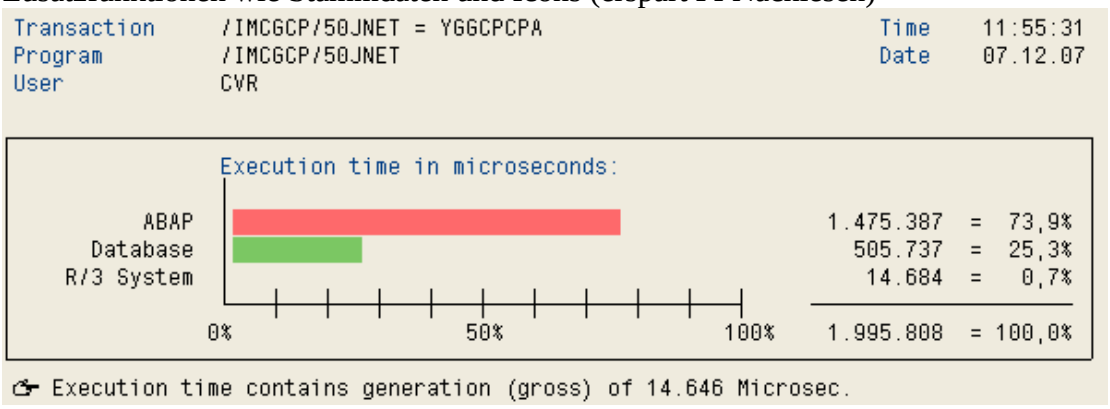
Ohne Visualisierung (Anzeigefunktion neu) => neu erstellte Anzeigefunktion mit deaktivierter Visualisierung



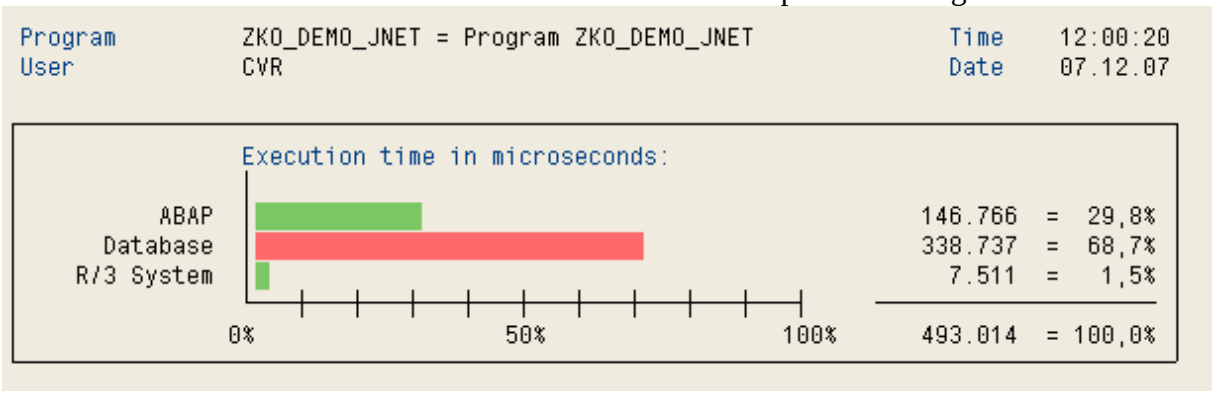
Mit Visualisierung & Nachlesen von FI und Stammdaten => Neue Anzeigefunktion mit vollem Funktionsumfang (Nachlesen aller Daten)



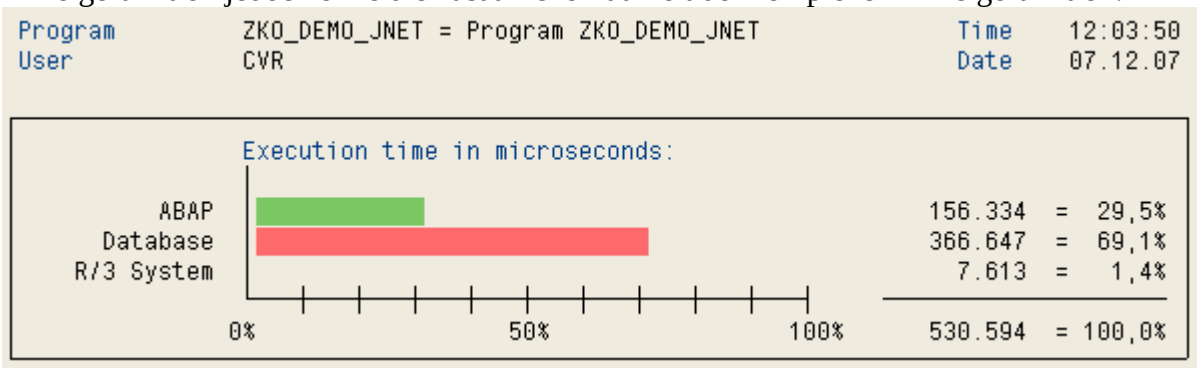
Mit Visualisierung, ohne Zusatzfunktion (Icons / Stammdaten) => Visualisierung ohne Zusatzfunktionen wie Stammdaten und Icons (erspart FI Nachlesen)



JNET Fenster in simpler Testapplikation (eingeschränkt) => Visualisierung ohne Zusatzdaten- und Funktionen in einer ohne Laufzeit der komplexen Anzeigefunktion

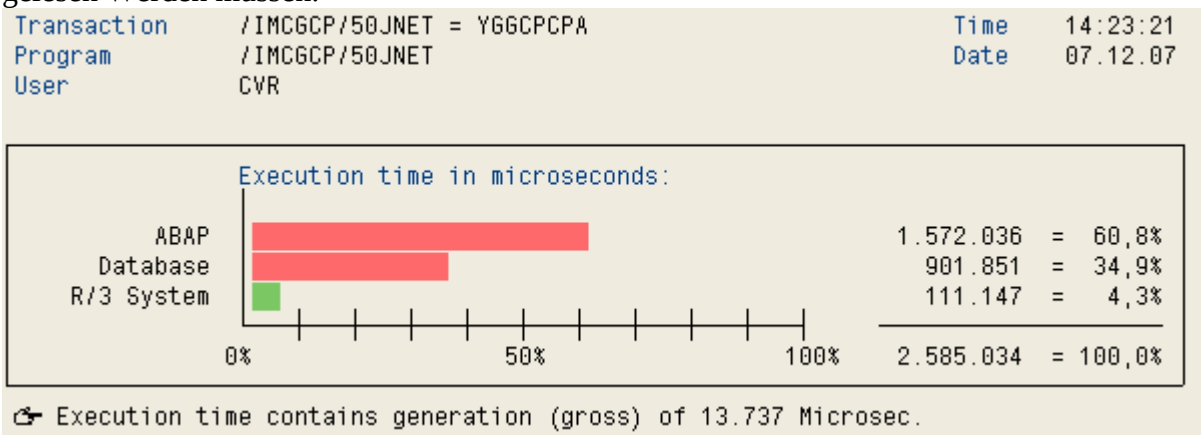


JNet Fenster in simpler Testapplikation (volle Funktionalität) => Volle Funktionalität der Anzeigefunktion jedoch ohne die zusätzliche Laufzeit der komplexen Anzeigefunktion.



Laufzeitmessung mit mehreren Navigationsschritten

=> In diesem Test wurde die Visualisierung gestartet und mehrmals der Kontext, durch einen Doppelklick, gewechselt. Die Laufzeit steigt bei mehreren Navigationsschritten nur unwesentlich an, da die meisten Daten schon im Puffer vorhanden sind, und nicht doppelt gelesen Werden müssen.

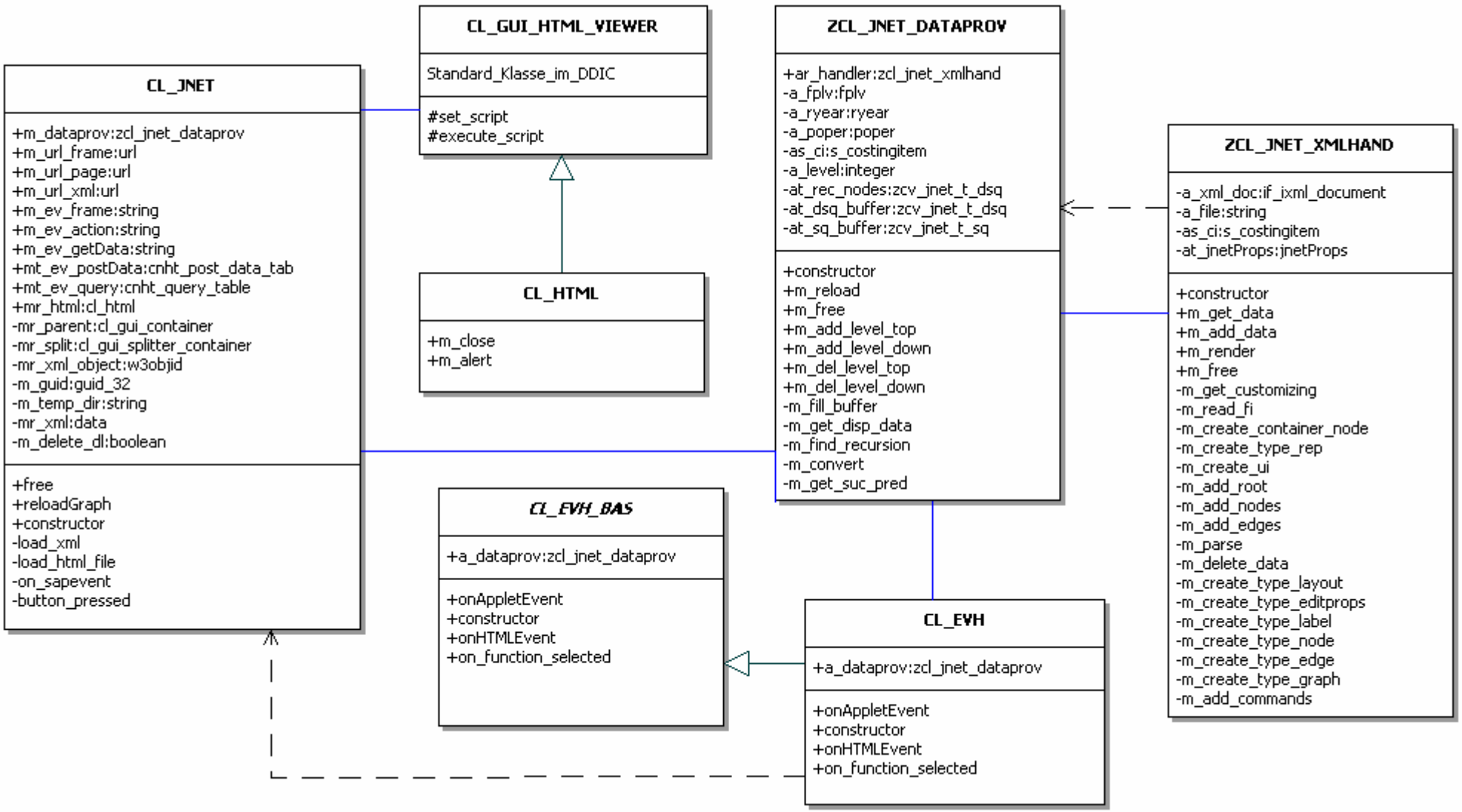


Anhang C

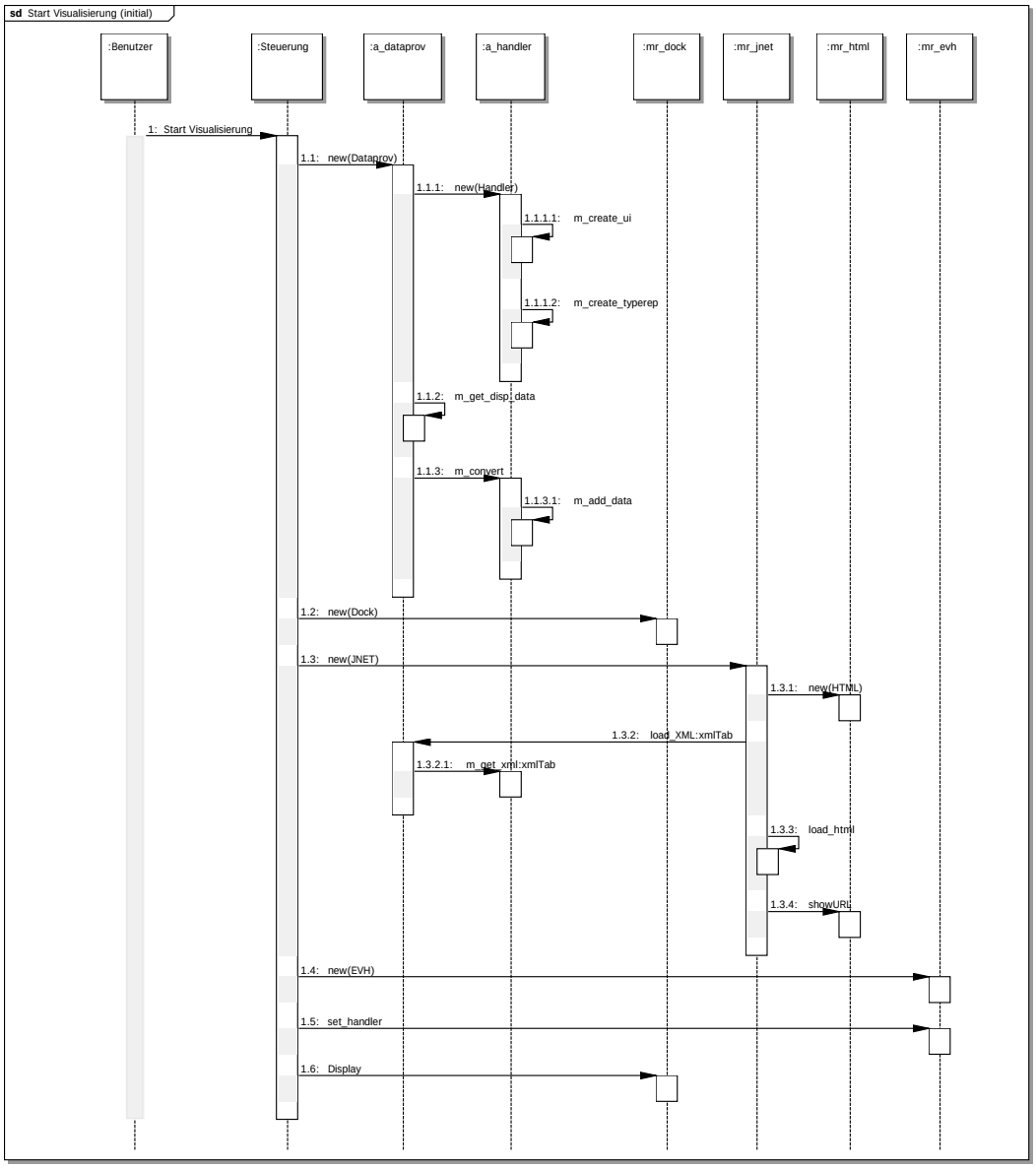
Klassendiagramm (Übersicht)

Sequenzdiagramme:

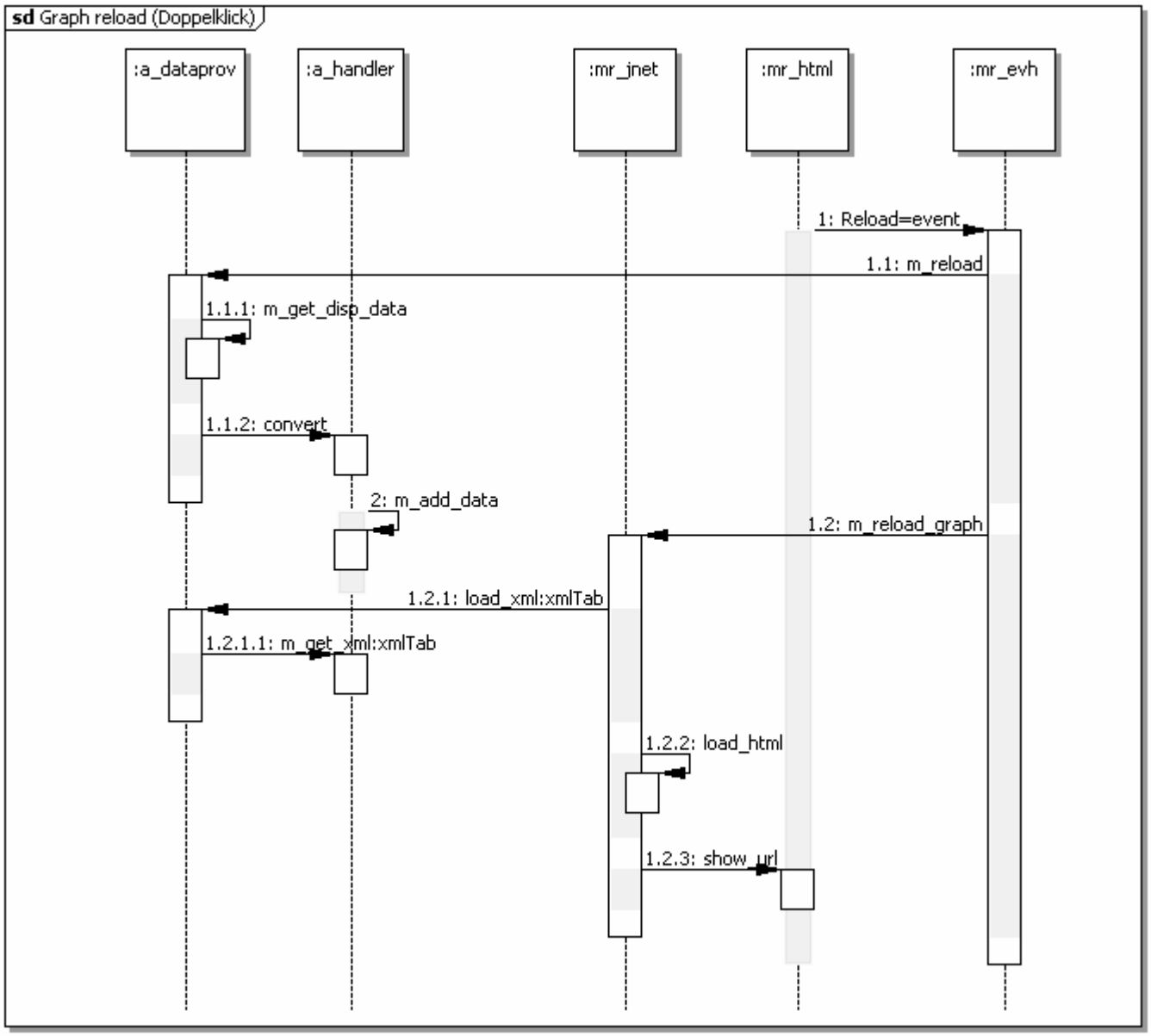
- Start der Visualisierung
- Doppelklick auf Knoten (Ereignisverarbeitung)


CL_GUI_HTML_VIEWER
Standard_Klasse_im_DDIC
#set_script
#execute_script
CL_JNET
+m_dataprov:zcl_jnet_dataprov
+m_url_frame:url
+m_url_page:url
+m_url_xml:url
+m_ev_frame:string
+m_ev_action:string
+m_ev_getData:string
+mt_ev_postData:cnht_post_data_tab
+mt_ev_query:cnht_query_table
+mr_html:cl_html
-mr_parent:cl_gui_container
-mr_split:cl_gui_splitter_container
-mr_xml_object:w3objid
-m_guid:guid_32
-m_temp_dir:string
-mr_xml:data
-m_delete_dl:boolean
+free
+reloadGraph
+constructor
-load_xml
-load_html_file
-on_sapevent
-button_pressed
CL_HTML
+m_close
+m_alert
ZCL_JNET_DATAPROV
+ar_handler:zcl_jnet_xmlhand
-a_fplv:fplv
-a_ryear:ryear
-a_poper:poper
-as_ci:s_costingitem
-a_level:integer
-at_rec_nodes:zcv_jnet_t_dsq
-at_dsq_buffer:zcv_jnet_t_dsq
-at_sq_buffer:zcv_jnet_t_sq
+constructor
+m_reload
+m_free
+m_add_level_top
+m_add_level_down
+m_del_level_top
+m_del_level_down
-m_fill_buffer
-m_get_disp_data
-m_find_recursion
-m_convert
-m_get_suc_pred
ZCL_JNET_XMLHAND
-a_xml_doc:if_ixml_document
-a_file:string
-as_ci:s_costingitem
-at_jnetProps:jnetProps
+constructor
+m_get_data
+m_add_data
+m_render
+m_free
-m_get_customizing
-m_read_fi
-m_create_container_node
-m_create_type_rep
-m_create_ui
-m_add_root
-m_add_nodes
-m_add_edges
-m_parse
-m_delete_data
-m_create_type_layout
-m_create_type_editprops
-m_create_type_label
-m_create_type_node
-m_create_type_edge
-m_create_type_graph
-m_add_commands
CL_EVH_BAS
+a_dataprov:zcl_jnet_dataprov
+onAppletEvent
+constructor
+onHTMLEvent
+on_function_selected
CL_EVH
+a_dataprov:zcl_jnet_dataprov
+onAppletEvent
+constructor
+onHTMLEvent
+on_function_selected

1.1, Sequenz Programmstart



1.1, Sequenz Programmstart


sd Graph reload (Doppelklick)
:a_dataprov
:a_handler
:mr_jnet
:mr_html
:mr_evh
1: Reload=event
1.1: m_reload
1.1.1: m_get_disp_data
1.1.2: convert
2: m_add_data
1.2: m_reload_graph
1.2.1: load_xml:xmlTab
1.2.1.1: m_get_xml:xmlTab
1.2.2: load_html
1.2.3: show_url

Anhang D

Quellcode der Applikation

CD mit elektronischer Version der Arbeit und SAP Tutor Demo

Installationsleitfaden