

Projektdokumentation

GS-Inventur

Bericht zur betrieblichen Projektarbeit von
Benjamin Wirtz
zur Erlangung des Abschlusses als
Fachinformatiker - Fachrichtung Anwendungsentwicklung

Betrieb:

Gandke & Schubert GmbH

Projektbetreuer:

Detlef Braun

Ausführungszeitraum:

14. bis 24. Oktober 2002

INHALTSVERZEICHNIS

1	THEMA DER PROJEKTARBEIT	1
1.1	Der Ausbildungsbetrieb und das Projektumfeld.....	1
1.2	Problemstellung und Beschreibung der Ausgangssituation	1
2	ABLAUF- UND RESSOURCENPLANUNG	2
2.1	Konzeptentwicklung	2
2.2	Ablaufplanung.....	2
2.3	Projektphasenplanung.....	3
2.4	Wirtschaftlichkeitsanalyse	3
2.4.1	Einzelkosten	3
2.4.2	Anteilige Gemeinkosten	4
2.4.3	Gesamtkosten	5
2.4.4	Ertrag.....	5
3	PLANUNG UND UMSETZUNG DER DATENBANK.....	6
3.1	Planung des ER – Modells.....	6
3.2	Umsetzung der Datenbank	7
4	IMPLEMENTATION DES PROGRAMMS	9
4.1	Programmierungsumgebung und Schnittstellen.....	9
4.2	Die Programmklassen.....	10
4.3	Erläuterung der Programmklassen und Funktionalität	10
4.3.1	Netzwerk nach Workstations durchsuchen	10
4.3.2	Durchsuchen der Workstations nach installierter Software	12
4.3.3	Die Schnittstelle zur Datenbank	13
4.4	Die grafische Benutzerschnittstelle.....	14
5	ABSCHLIEßENDE TESTS DER ANWENDUNG	18
6	FAZIT.....	19
7	ANHANG A DER PROJEKTAUFTRAG.....	20
8	ANHANG B DAS KLASSENDIAGRAMM.....	22
9	ANHANG C DER SOURCECODE	23
10	ANHANG D DAS PFLICHTENHEFT.....	30

1 Thema der Projektarbeit

1.1 Der Ausbildungsbetrieb und das Projektumfeld

Die Firma Gandke & Schubert GmbH entwickelt seit 14 Jahren kaufmännische Standard- und Branchensoftware und ist mit rund 670.000 installierten Versionen und weit über 260.000 Anwendern einer der Marktführer in diesem Bereich. Zurzeit beschäftigt Gandke & Schubert ca. 45 Mitarbeiter.

1.2 Problemstellung und Beschreibung der Ausgangssituation

In der Gandke & Schubert GmbH werden die verschiedensten Softwareprodukte auf verschiedenen Arbeitsstationen für die Bewältigung der täglichen Arbeit eingesetzt. Für diese Softwareprodukte müssen immer ausreichend Softwarelizenzen vorhanden sein, um das Unternehmen vor der Übertretung des UrhG (Urheberrechtsgesetz) zu schützen.

Zur Zeit werden die PC sowie die Softwarelizenzen in der Buchhaltung erfasst, verwaltet und ausgewiesen. Eine Zusammenführung von Rechnern und Lizenzen und die Inventur werden quartalsweise manuell vorgenommen.

Zur Automatisierung der Inventur von Software, die auf den einzelnen Workstations und Servern im Firmennetzwerk der Gandke & Schubert GmbH & Co. KG installiert ist, soll im Auftrag der Geschäftsführung eine Software entwickelt werden. Es soll lizenzierungspflichtige Software wie z.B. Betriebssysteme, Microsoft Office Produkte und die Entwicklungsumgebungen erfasst werden. Ebenso ist die Möglichkeit einer Auswertung zu realisieren. Diese soll aufzeigen, welche Lizenzen vorhanden sind, welche eingesetzt werden und wo eventuell Differenzen auftreten.

Durch das Programm muss es überprüfbar sein, ob die im Unternehmen eingesetzte Software immer nachweisbar lizenziert ist. Weiterhin sollen Kostensparpotenziale realisiert werden, die durch die manuelle Verwaltung von Lizenzen, durch die manuelle Inventur und manuell editierten Inventurprotokolle anfallen.

Das Programm soll der Geschäftsführung aufzeigen, auf welcher Arbeitsstation im Firmennetzwerk die Programme installiert sind und welche Lizenzen diesen Programmen zugeordnet sind. Die Daten werden in einer zentralen Datenbank verwaltet.

Der Geschäftsführer kann zu beliebigen Zeitpunkten verschiedene Auswertung generieren. Hierbei soll erkennbar werden, welche Lizenzen vorhanden sind, welche durch den Einsatz von Software bereits vergeben sind oder ob nachlizenziert werden muss. In der Verwaltung muss es außerdem möglich sein, sowohl PCs als auch Software und Lizenzen manuell einzupflegen oder zu editieren.

Zum Zeitpunkt der Auswertung durchsucht die Anwendung das Netzwerk nach vorhandenen Rechnern und ermittelt lizenzierungspflichtige Software wie Betriebssysteme, Microsoft Office Produkte und die benötigten Entwicklungsumgebungen in der Windows – Registry.

Diese Informationen werden in der Datenbank gespeichert. Anhand dieser Daten können dann Ist/Soll Auswertungen erstellt werden und die Ergebnisse als Protokolle ausgegeben werden.

2 Ablauf- und Ressourcenplanung

2.1 Konzeptentwicklung

Anhand der oben beschriebenen Ausgangssituation und des Projektauftrages, der unter Anhang A beiliegt, entwickelte ich das Projektkonzept, welches sich in den folgenden Darstellungen des Ablaufplans und des Projektphasenplans niederschlägt. Ebenso hielt ich das Soll – Konzept in Form eines Pflichtenheftes, welches unter Anhang D beiliegt, fest um die Vereinbarungen mit dem Auftraggeber zu dokumentieren.

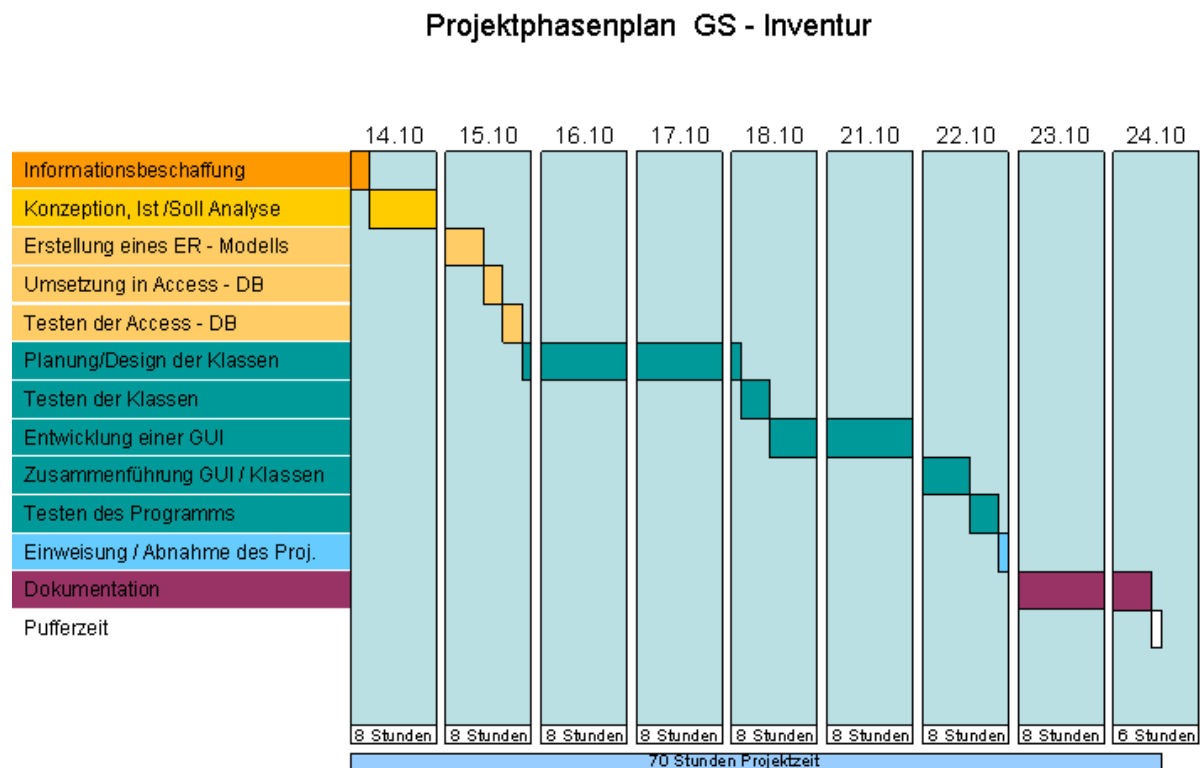
2.2 Ablaufplanung

Die Ablaufplanung zeigt eine detaillierte Aufschlüsselung der einzelnen Projektphasen in zeitliche Abschnitte.

	Prozesse	Prozessdauer	kumulierte Dauer
1	Informationsbeschaffung für die Durchführung des Projektes	2 Std.	2 Std.
2	Konzeptentwicklung, Ist-Analyse/ Soll – Konzept, Pflichtenheft, Wirtschaftlichkeitsanalyse.	6 Std.	8 Std.
	Datenbank erstellen	7 Std.	15 Std.
3	Erstellen des ER - Modells	3 Std.	
4	Umsetzung des ER - Modells in eine Accessdatenbank + Datenimport der Lizenzdaten aus der Buchhaltung	2 Std.	
5	Testen der Accessdatenbank	2 Std.	
	Entwicklung des Programms	40 Std.	55 Std.
6	Entwicklung der Programmklassen	18 Std.	
7	Testen des Klassen	3 Std.	
8	Entwickeln einer graf. Benutzerschnittstelle (GUI)	12 Std.	
9	Zusammenführung des GUI und der Klassen	4 Std.	
10	Testen des Programms	3 Std.	
	Präsentation	1 Std.	56 Std.
11	Präsentation und Einweisung der Geschäftsführung / Abnahme des Projektes	1 Std.	
12	Dokumentation	12 Std.	68 Std.
13	Pufferzeit	2 Std.	70 Std.

2.3 Projektphasenplanung

Der Projektphasenplan ist eine Übersicht über die Verteilung der einzelnen Projektphasen über die gesamte Projektdauer hinweg.



Durchführungszeitraum: 14.10 – 24.10.2002

Benjamin Wirtz

2.4 Wirtschaftlichkeitsanalyse

2.4.1 Einzelkosten

Da bei der Auswahl eines geeigneten Tools für eine Inventur auch auf die individuelle Anpassung an die internen Gegebenheiten bei Gandke & Schubert (zwecks Datenübernahme und Anbindung an Verwaltung) geachtet werden musste kam ein externes Tool aus Kostengründen nicht in Frage.

Die Lohnkosten und Lohnnebenkosten für die Durchführung des 70stündigen Projekts betragen:

Ausbildungsvergütung & VWL		230,42€
Rentenversicherung	+	27,42€
Arbeitslosenversicherung		9,33€
Pflegeversicherung	+	2,15€
Krankenversicherung	+	17,80€
Fahrkostenzuschlag & Unfallversicherung	+	32,45€
<hr/>		
Kosten	=	319,57 €

Aufschlüsselung der Kosten:

Sozialversicherungspflichtig (Ausbildungsvergütung, VWL (Vermögenswirksame Leistungen)):

Die 220 Arbeitstage setzen sich zusammen aus:

reale Arbeitstagen pro Jahr – Urlaubstage pro Jahr – Krankheitstage pro Jahr = durchschnittlich 220 Arbeitstage pro Jahr

220 Arbeitstage * 8 Std. pro Tag = 1760 Std. im Jahr

versicherungspflichtiges Einkommen pro Monat		555,30€
Monate (Januar-Dezember + Weihnachtsgeld)		*13
versicherungspflichtiges Einkommen im Jahr	=	7218,90 €

Ausbildungsvergütung und VWL für 70 Std. = $7218,90 \text{ €} / 1760 * 70$ = 287,12 €

Darin enthalten sind die Arbeitgeberanteile der:

Rentenversicherung (9,55%)	=	27,42 €
Arbeitslosenversicherung (3,25%)	=	9,33 €
Pflegeversicherung (0,75%)	=	2,15 €
Krankenversicherung (6,2%)	=	17,80 €

Nicht versicherungspflichtig (Fahrkostenzuschlag, Unfallversicherung):		
Fahrkostenzuschlag 62,00 €* 13 Monate	=	806,00 €
Zwischenergebnis 806,00 €+ 10 €Unfallversicherung	=	816,00 €
Ergebnis 816,00 €/ 1760 Std. * 70 Std.	=	32,45 €

2.4.2 Anteilige Gemeinkosten

In den Gemeinkosten werden unter anderen folgende Positionen erfasst:

Raummiete
Stromverbrauch
Heizkosten
Büroverbrauchsmaterialien
etc. ...

Die Ermittlung der anteiligen Gemeinkosten ist im Rahmen dieser Projektarbeit nicht möglich. Es wird jedoch ein kalkulatorischer Pauschalbetrag von **25 €** erhoben.

2.4.3 Gesamtkosten

Einzelkosten + Anteilige Gemeinkosten = Gesamtkosten

319,57 € + 25 € = **344,57 €**

Die Gesamtkosten betragen **344,57 €**

2.4.4 Ertrag

Um Ertragswerte zu ermitteln habe ich unter Punkt 2.4.4 den Ertrag im Vergleich zur externen Entwicklung und unter Punkt 2.4.5 den Ertrag im Vergleich Azubi und Angestellter aufgeschlüsselt.

2.4.4.1 Ertrag im Vergleich zur externen Entwicklung

Gemessen an den GS – Stundensätzen hätte sich bei einer 70 stündigen Individualentwicklung folgende Kosten ergeben:

GS – Stundensatz = 75 € * 70 Stunden

Gesamtkosten externe Individualentwicklung	5250 €
Gesamtkosten Azubiprojekt	344,57 €
Ertrag	4905,43 €

Somit ergibt sich bei interner Entwicklung ein Ertragswert von 4905,43 € Ebenso werden weiter Kostenspareffekte realisiert, da die sonst manuell durchgeführte Inventur der im Hause installierten Software weitgehend automatisiert abläuft.

Außerdem kann die Geschäftsleitung auf Basis der Auswertungen dieses Programms die Inventur der installierten Software im Hause schneller erfassen und besser beurteilen, bewerten und steuern und so möglicherweise künftig Kostenspareffekte realisieren.

2.4.4.2 Alternative Ertragsermittlung für festangestellte Entwickler

Bei einer alternativen Kostenaufstellung für einen festangestellten Entwickler mit einem versicherungspflichtigem Einkommen von 2500€ ergibt sich folgende Rechnung:

Angenommen werden: 220 Arbeitstage * 8 Std. pro Tag = 1760 Std. im Jahr

versicherungspflichtiges Einkommen pro Monat	2500,00 €
Monate (Januar-Dezember + Weihnachtsgeld)	*13
versicherungspflichtiges Einkommen im Jahr	= 32500,00 €

Daraus ergibt sich für ein 70 stündiges Projekt :

$$32500,00 \text{ €} / 1760 * 70 = 1292,61 \text{ €}$$

Darin enthalten sind die Arbeitgeberanteile der:

Rentenversicherung (9,55%)	= 123,44 €
Arbeitslosenversicherung (3,25%)	= 42,00 €
Pflegeversicherung (0,75%)	= 9,69 €
Krankenversicherung (6,2%)	= 80,14 €

Nicht versicherungspflichtig (Fahrkostenzuschlag, Unfallversicherung):		
Fahrkostenzuschlag	62,00 €* 13 Monate	= 806,00 €
Zwischenergebnis	806,00 €+ 10 €Unfallversicherung	= 816,00 €
Ergebnis	816,00 €/ 1760 Std. * 70 Std.	= 32,45 €

Auch hier wird für die anteiligen Gemeinkosten ein kalkulatorischer Pauschalbetrag von **25 €** erhoben.

Somit ergeben sich folgende Gesamtkosten :

Einzelkosten + Anteilige Gemeinkosten = Gesamtkosten

1292,61 €+ 25 €= **1317,61 €**

Ertrag:

Gesamtkosten festangestellter Entwickler 1317,61 €

Gesamtkosten Azubiprojekt 344,57 €

Ertrag 973,04 €

3 Planung und Umsetzung der Datenbank

3.1 Planung des ER – Modells

Zur Strukturierung und Erfassung der zu verwaltenden Daten wird eine Microsoft Access Datenbank eingesetzt, die von der Anwendung über die ADO – Schnittstellen angesprochen wird.

Für die Erstellung des Entity-Relationship-Modells (ER-Modell) analysierte ich zunächst, welche Datenobjekte (Entitäten) in der Datenbank erfasst werden müssen. Dabei erwiesen sich folgende Objekte als sinnvoll:

Die Workstation als Repräsentant für die einzelnen Arbeitsstationen. Diese können über die IP – Adresse und den Workstationnamen eindeutig identifiziert werden. Den Arbeitsstationen können dann die einzelnen Softwareprodukte eindeutig zugeordnet werden.

Das Softwareprodukt, welches die einzelnen auf den Rechnern installierten Produkte darstellt. Hier wird der Softwarename gespeichert der eine Identifizierung in der Registry zulässt. Ebenso wird dem einzelnen Produkt eine ID zugewiesen um so eine eindeutige Zuordnung in der Datenbank zu ermöglichen.

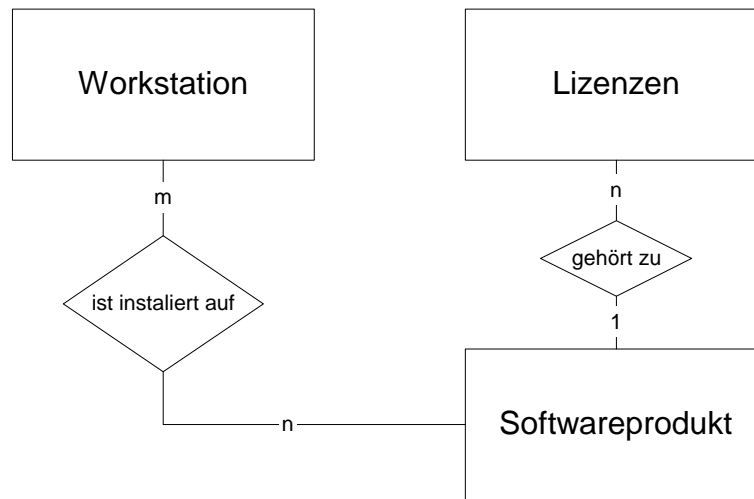
Die Lizenz. Die Lizenz wird repräsentiert durch die in der Buchhaltung verwalteten Software - Rechnungen. Hier soll später jedem Softwareprodukt eine Rechnung zugeordnet werden können.

Die Entitäten sind über Beziehungen miteinander verknüpft und ergeben so das ER-Modell. Die einzelnen Verknüpfungen sehen wie folgt aus:

Auf jeder Workstation können verschiedene Softwareprodukte installiert sein, jedoch können auch die verschiedenen Softwareprodukte auf unterschiedlichen Workstations installiert sein. Daraus ergibt sich eine „m : n“ Beziehung die in der Datenbank als zusätzliche Tabelle realisiert wird. Die Beziehung zwischen den Rechnungen und den

Softwareprodukten ist eine „1 : n“ Beziehung, da für ein Softwareprodukt mehrere Rechnungen vorhanden sein können.

Somit stellt sich das ER – Modell wie folgt dar:



3.2 Umsetzung der Datenbank

Anhand des nun vorliegenden ER – Modells wird die Access Datenbank „Lizenz.mdb“ erstellt. Hierzu werden die einzelnen Entitäten als Tabellen realisiert. Wie bereits oben erwähnt wird für die „m : n“ Beziehung zwischen den Entitäten Workstation und Softwareprodukt eine zusätzliche Zwischentabelle angelegt über diese die Beziehung realisiert wird.

Die sich aus den Entitäten ergebenden Attribute werden in den Tabellen als Felder realisiert. Die einzelnen Felder der Tabellen werden im Einzelnen kurz dargestellt.

Workstation -> Tworkstation:

Key:	Feldname:	Datentyp:	Größe:
PK	TworkstationID	AutoWert	Long Integer
	IPAdresse	Text	50
	MacAdresse	Text	50
	ClientName	Text	50

Softwareprodukte -> TinstalledSoftware:

PK	TSoftwareID	AutoWert	Long Integer
	ProduktName	Text	50

Lizenzen -> TblSoftDet:

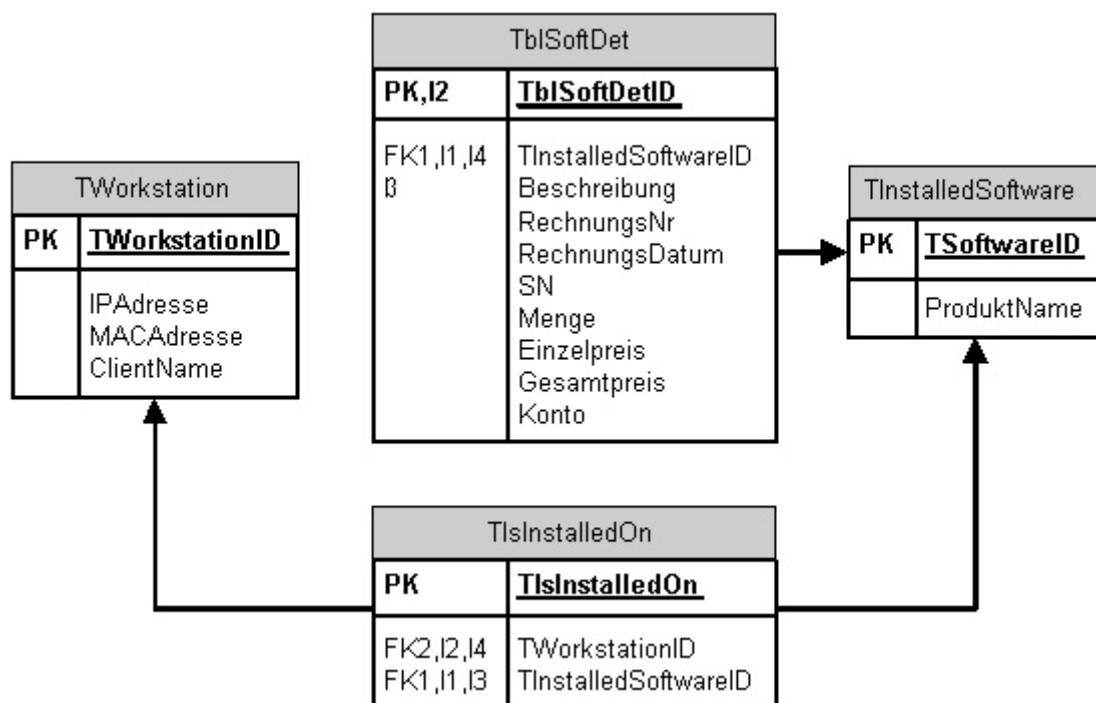
PK	TblSoftDetID	AutoWert	Long Integer
FK	TInstalledSoftwareID	Zahl	Long Integer
	Beschreibung	Text	255
	RechnungNr	Text	12
	RechnungsDatum	Datum	
	SN	Text	25
	Menge	Zahl	Integer
	Einzelpreis	Währung	Euro
	Gesamtpreis	Währung	Euro
	Konto	Text	8

Kreuztabelle -> TIsInstalledOn:

PK	TIsInstalledOn	AutoWert	Long Integer
FK	TWorkstation	Zahl	Long Integer
FK	TInstalledSoftwareID	Zahl	Long Integer

PK.: Primary Key FK.: Foreign Key

Daraus ergibt sich folgendes relationales Datenbankmodell:



Zur Vermeidung unerwünschter Datenredundanzen und Updateanomalien habe ich die Datenbank normalisiert. Die Datenbank befindet sich in der dritten Normalform.

Die Normalformen:

1. Eine Entität (Tabelle) befindet sich in der ersten Normalform, wenn alle Attribute nur einen einzigen Wert besitzen.
2. Eine Entität befindet sich in der zweiten Normalform, wenn sie bereits in der ersten Normalform vorliegt und alle nichtidentifizierenden Attribute von der eindeutigen ID der Entität abhängen.
3. Eine Entität befindet sich in der dritten Normalform, wenn sie bereits in der zweiten Normalform vorliegt und kein nichtidentifizierendes Attribut von einem anderen nichtidentifizierenden Attribut abhängig ist.

Entität (Tabellenname):

Eine Entität stellt einen Themenkreis dar, welcher Elemente mit gleichen Merkmalen umfasst.

Tabelle:

Eine Tabelle umfasst eine Entität mit der dazugehörenden Entitätsmenge. Man versteht darunter eine komplette Tabelle mit Entitätsbezeichnung, Attributen und Tupel.

Attribut (Spaltenname):

Das Attribut entspricht einem Merkmal eines Tupels und beschreibt somit eine spezifische Eigenschaft einer Entitätsmenge.

(Quelle: MYSQL & mySQL, O'REILLY, Kapitel Datenbank-Design, Seite 17 ff.)

4 Implementation des Programms

4.1 Programmierumgebung und Schnittstellen

Für die Implementierung der Anwendung „GS – Inventur“ wurde die Entwicklungsumgebung Delphi 5 Enterprise von Borland verwendet. Delphi 5 ist ein RAD – Tool (Rapid Applikation Development), mit dem sich schnell komplexe Anwendungen realisieren lassen.

Die von Delphi verwendete Sprache ist Objekt Pascal, welche die Eigenschaften der OOP (Objektorientierten Programmierung) vollständig umsetzt und somit die Verwendung eines OOP – Konzeptes zulässt.

Zusätzlich werden in der Programmierung bei Gandke & Schubert die Softwarekomponenten der Firma „Dream Company“ eingesetzt. Für diese liegt eine Volumenlizenz vor, so dass jeder Entwickler diese in seinen Anwendungen einsetzen kann.

Die Kommunikation zwischen Anwendung und Datenbank wird über eine eigens dafür implementierte Klasse realisiert, welche sich der ADO – Komponenten (ActiveX DataObjects) der Delphi VCL (Virtual Class Library) bedient.

Zugriffe auf die Netzwerkinfrastruktur werden über die von der Windows API (Application Programmers Interface) zur Verfügung stehenden Funktionen realisiert.

4.2 Die Programmklassen

Zur Vereinfachung der Übersicht der von mir entworfenen Klassen und deren Zusammenspiel, stelle ich diese in einem Klassendiagramm dar. Zu erkennen sind die einzelnen Felder und Methoden der Klassen und deren Beziehung untereinander.

Das Modell zeigt die Klassen als Rechtecke, bei denen der Klassenname gelb hinterlegt wird. Der Untere Bereich ist in die beiden Bereiche „attributes“, in dem die Felder der Klasse definiert werden und „operations“, in dem die Methoden der Klasse definiert sind aufgeteilt. Die einzelnen Assoziationen der Klassen werden als Pfeile dargestellt, wobei immer der Pfeil auf die Klasse zeigt die von der aufrufenden Klasse referenziert wird.

Zur Vereinfachung erläutere ich nur die Klassen und Funktionen, die von der Funktionalität in direkter Beziehung zueinander stehen und für den Programmablauf direkt von einander abhängen. Das gesamte Klassendiagramm ist im Anhang B zu sehen, hier ist auch eine kurze Legende zur Erläuterung zu sehen.

4.3 Erläuterung der Programmklassen und Funktionalität

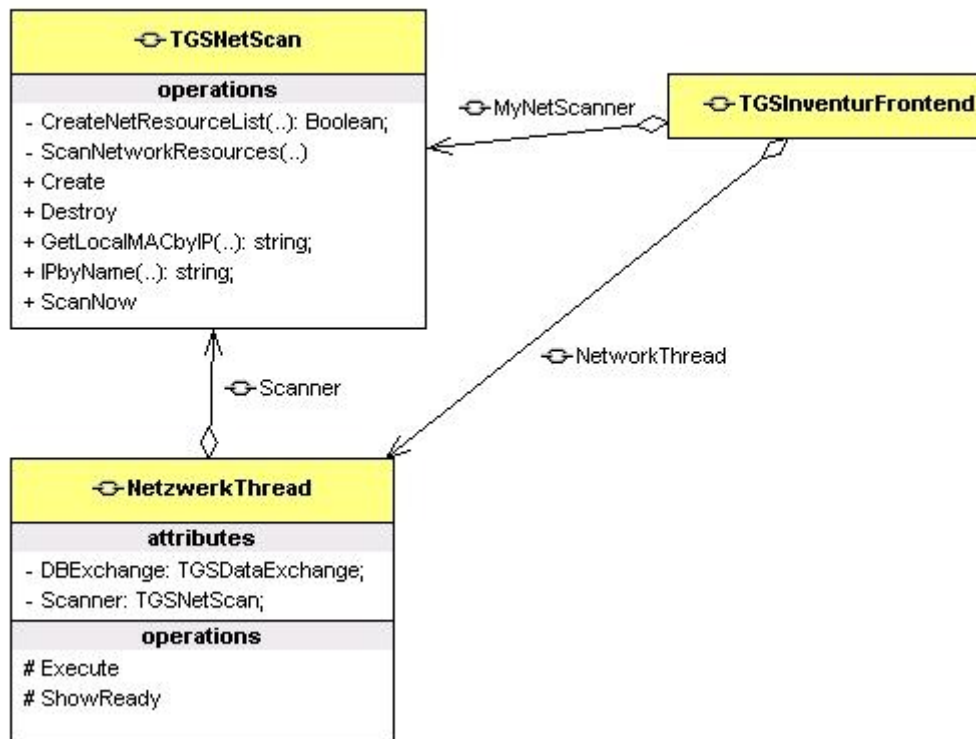
4.3.1 Netzwerk nach Workstations durchsuchen

Die Klasse TGSInventurFrontend stellt die Hauptklasse in dem Projekt dar. Sie repräsentiert das Grafische User Interface (GUI) der Anwendung. Diese Klasse wird beim Starten des Programms aufgerufen und führt in ihrer ONCreate – Methode alle weiteren Funktionen aus oder wartet auf eine Eingabe des Benutzers.

Um festzustellen wieviele und welche Clients im Netzwerk vorhanden sind, muss beim Start der Anwendung das Netzwerk durchsucht werden. Da dieser Prozess etwas länger dauert, jedoch dem Benutzer von Anfang an alle Programmfunktionalitäten zur Verfügung stehen sollen, habe ich diesen Vorgang in einen Thread ausgelagert.

Ein Thread ist ein parallel zur Hauptanwendung laufender Prozess, der gleichzeitig ausgeführt wird. Das Betriebssystem teilt die CPU-Zeit für die Prozesse auf.

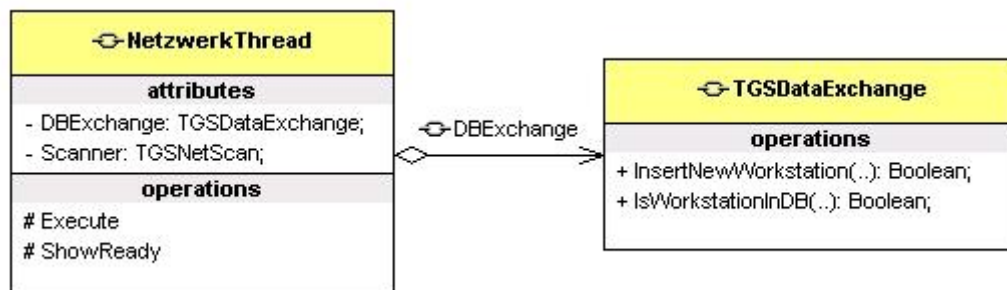
Der folgende Ausschnitt aus dem Klassendiagramm zeigt die Klasse TGSInventurFrontend. Diese ruft in ihrer OnCreate Methode einen Thread auf der in der Klasse „NetzwerkThread“ implementiert ist. In der Methode „Execute“ der Klasse „NetzwerkThread“ ist die Funktionalität programmiert, die während des Threads abläuft.



Hier wird die Klasse „TGSNetScan“ aufgerufen, die vom Typ „TStringList“ abgeleitet ist. Das bedeutet dass diese Klasse eine Liste vom Typ String repräsentiert. In der Methode "ScanNow" wird das Netzwerk nach Clients durchsucht und die gefundenen Workstationnamen in die Stringliste geladen.

Die suche nach den einzelnen Workstations im Netzwerk wird über die Windows-API „WNetOpenEnum“ realisiert. Diese liefert eine Aufzählung von Netzwerkressourcen und offenen Verbindungen.

Um den Namen der Workstation in die dazugehörige IP aufzulösen hat die Klasse „TGSInventurFrontend“ auch direkten zugriff auf die Klasse „TGSNetScan“. Hier ist die Funktion „IpbyName“ implementiert, die als Übergabeparameter den Workstationnamen als String erhält und dann die IP der Workstation als String zurückliefert.

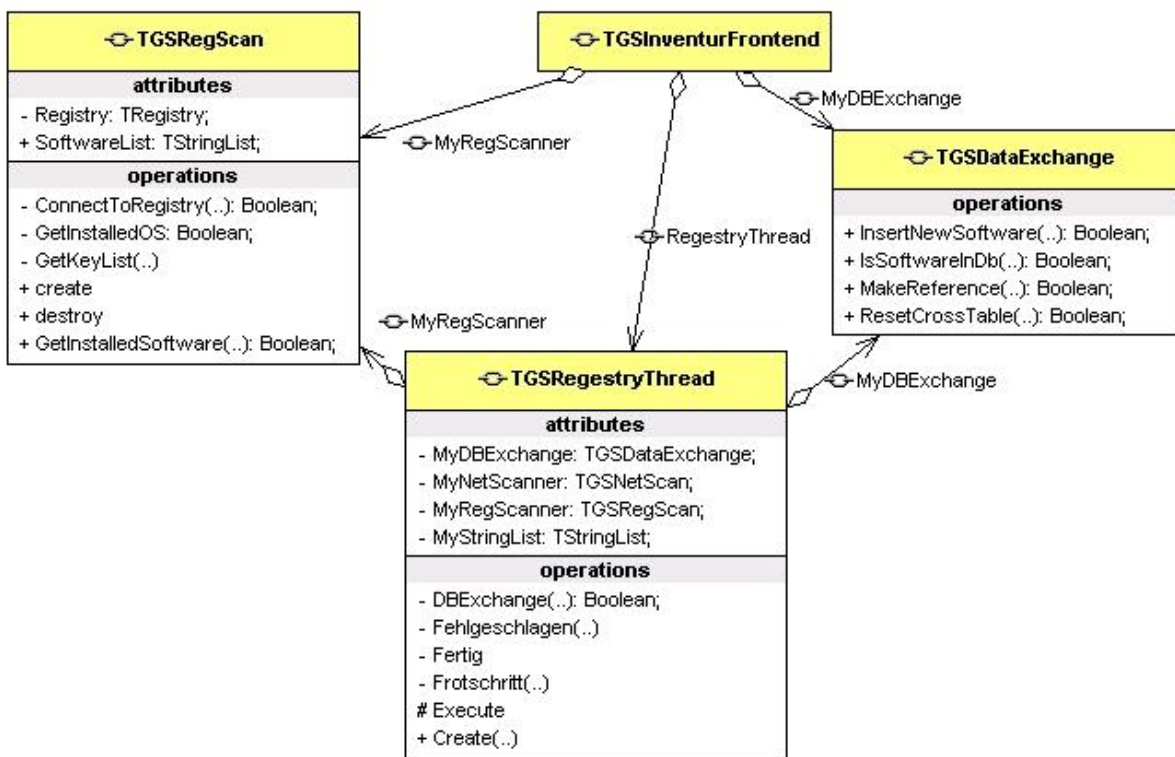


Nachdem die Suche beendet ist werden die gefundenen Workstations über die Funktion „InsertNewWorkstation“ der Klasse „TGSDataExchange“ in die Datenbank eingefügt.

4.3.2 Durchsuchen der Workstations nach installierter Software

Nachdem die Liste der im Netzwerk gefundenen Clients in der Datenbank gespeichert ist, werden diese auch in einem TreeView in der Anwendung angezeigt. Hier kann der Anwender die einzelnen Workstations auswählen und nähere Informationen wie z.B. installierte Software und IP-Adresse über diese angezeigt bekommen.

Wird vom Benutzer die Inventur angestoßen, wird auf den einzelnen Clients in der Registry nach eingetragenen Softwareprodukten gesucht. Dies geschieht ebenfalls über einen Thread. Der „TGSRegistryThread“ ruft in seiner „Execute“ Methode die Klasse „TGSRegScan“ auf und übergibt dieser den Namen der zu durchsuchenden Workstation.



Diese durchsucht in der Methode „GetInstalledSoftware“ die Registry im Schlüssel „HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall“ nach installierter Software. Die Produktnamen der gefundenen Software werden in die String-Liste „SoftwareList“ eingetragen auf die von außen zugegriffen werden kann.


Die Softwareliste wird an die Klasse „TGSDataExchange“ übergeben und die noch nicht in der Datenbank vorhandenen Produkte über die Funktion „InsertNewSoftware“ in die Daten eingetragen. Da hier auch die Kreuztabelle zwischen den Tabellen „TWorkstation“ und „TInstalledSoftware“ aktualisiert wird, muss diese zuerst über die Funktion „ResetCrossTable“ an den entsprechenden Stellen zurückgesetzt werden und dann über die Funktion „MakeReference“ die neuen Referenzen eingefügt werden. Dies geschieht in einer Datenbank – Transaktion, welche verhindert, dass die Daten bei Fehlern beschädigt werden oder verloren gehen. Tritt ein Fehler während der Transaktion auf, werden die veränderten Daten in ihren Urzustand zurückversetzt.

4.3.3 Die Schnittstelle zur Datenbank

In der Klasse "TGSDDataExchange" ist die Grundsätzliche Datenbankfunktionalität implementiert. Die Klasse kann über die Methode „ConnectDatabase“ die Verbindung zur Datenbank aufrufen. Hierzu wird der Methode ein so genannter Connection-String übergeben, in dem u.a. Informationen über den Pfad, den Typ der Datenbank enthalten sind. Ruft man die Überladene „ConnectDatabase“ Methode auf, welche keinen Connection-String benötigt, durchsucht die Anwendung das Verzeichnis in dem sie installiert ist nach der Datenbank „Lizenz.mdb“ und Trägt den Pfad sowie Datenbanktyp in der Registry ein. Die Verbindung wird so automatisch hergestellt.

Des weitere implementiert die Klasse die Methoden die für die Anwendung benötigt werden um grundsätzliche Aufgaben in der Datenbank zu erledigen. Wie z.B. Das Einfügen einer neuen Workstation.

Beispiel eines Connection-Strings:

 TGSDDataExchange
attributes
- fAdoConnection: TADOConnection; - fAdoDataSet: TADODataSet; - fAdoQuery: TADOQuery; + DataSet: TADODataSet;
operations
- GetDataSet: TADODataSet; - SetDataSet(..) + Create + Destroy + ConnectDatabase: Boolean; + ConnectDatabase(..): Boolean; + DisconnectDB: Boolean; + GetDataSetFromDB(..): Boolean; + GetSoftwareFromWorkstation(..): Boolean; + GetValidSoftwareList(..): Boolean; + GetWksIP(..): string; + GetWksMac(..): string; + InsertNewSoftware(..): Boolean; + InsertNewWorkstation(..): Boolean; + IsSoftwareInDb(..): Boolean; + IsWorkstationInDB(..): Boolean; + MakeReference(..): Boolean; + ResetCrossTable(..): Boolean; + ReturnDataSetFromDB(..): Boolean; + ReturnSoftwareID(..): string; + ReturnWorkstationID(..): string; + SetDataToDB(..): Boolean;

„Provider=Microsoft.Jet.OLEDB.4.0;DataSource=C:\DokumenteundEinstellungen\bwirtz\Eigenes Projekt\Sourcen\Lizenz.mdb;Persist Security Info=False“

Beispiel einer Datenbankfunktion:

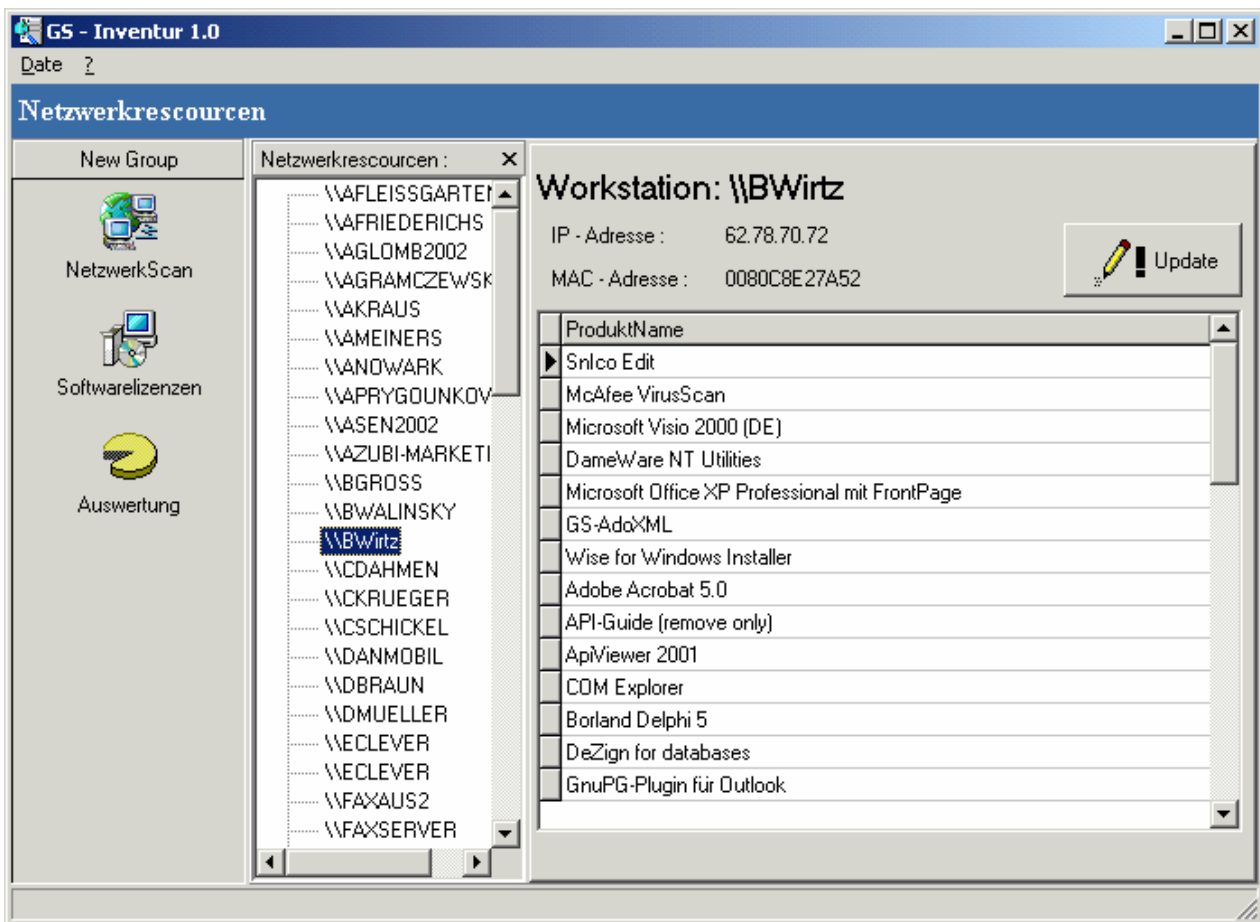
Hier wird die Softwareliste aus der Datenbank geladen.

```
function TGSDDataExchange.GetValidSoftwareList(aAdoDataSet: TADODataSet):  
    Boolean;  
  
var  
    SQL: string;  
  
begin  
    SQL:='SELECT * FROM TInstalledSoftware ORDER BY TInstalledSoftware.ProduktName;';  
    result:=False;  
    try  
        ReturnDataSetFromDB(SQL,aAdoDataSet);  
    except  
        result:=False;  
        exit;  
    end;  
    result:=TRUE;  
end;
```


4.4 Die grafische Benutzerschnittstelle

Um dem Benutzer eine möglichst übersichtliche Oberfläche zu bieten habe ich mich am Vorbild von Microsoft Outlook orientiert. Hierzu verwendete ich die Dream Komponenten der Firma Dream-Company (www.dream-com.com), mit deren Hilfe sich eine Anwendung im Outlook-Stile schnell realisieren lässt.

In der Abbildung unten ist die Anwendung „GS-Inventur“ dargestellt. Es ist die Baumansicht des lokalen Netzwerkes angezeigt und auf der rechten Seite die auf der ausgewählten Workstation installierte Software. Außerdem werden Informationen wie Workstationname, IP-Adresse und Mac-Adresse der jeweiligen Arbeitsstation angezeigt.

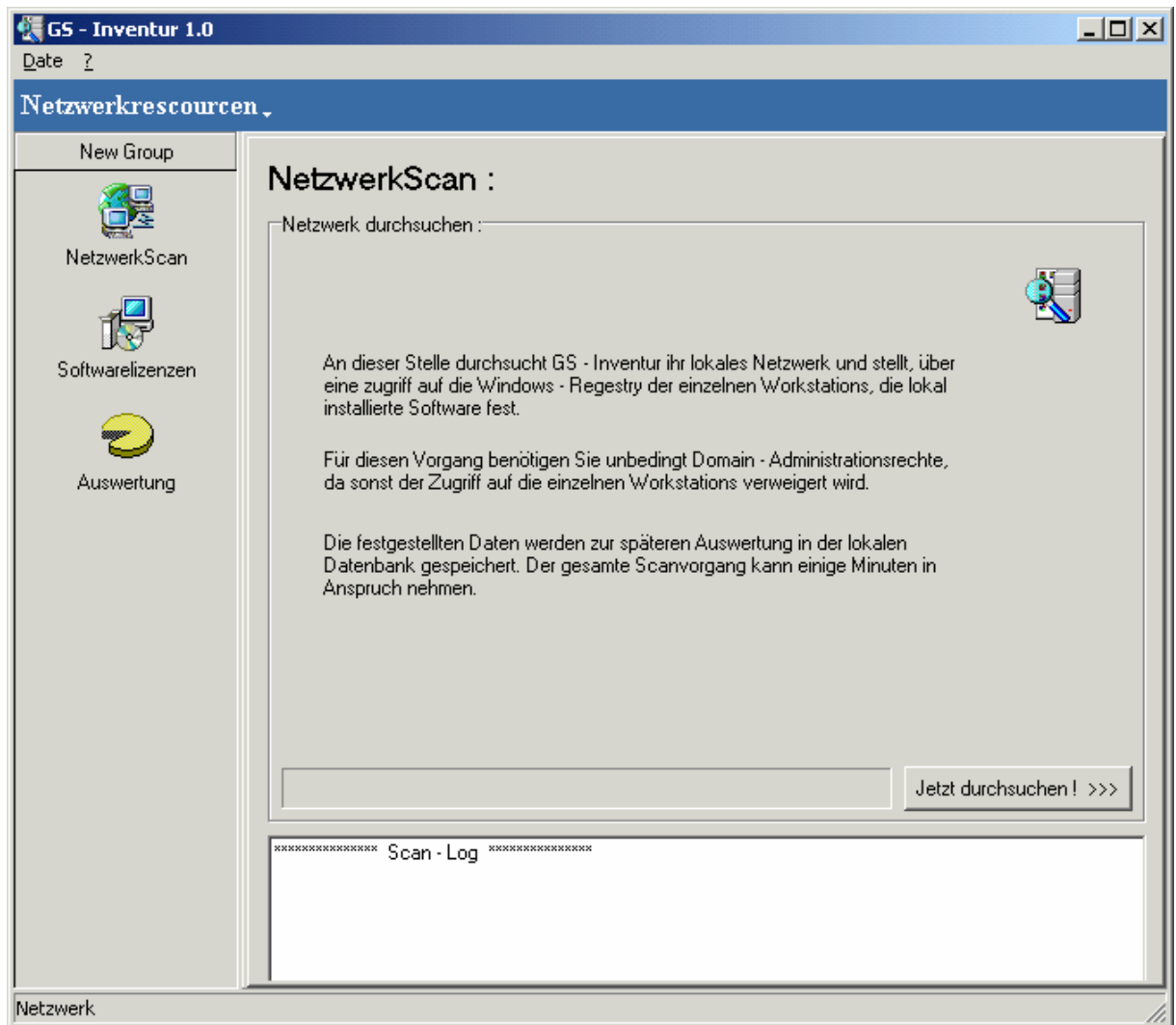


Der Benutzer kann über die drei Programmgruppen die in der linken Gruppenauswahl zur Verfügung stehen zu den einzelnen Programmfunktionen jederzeit wechseln.

Hier stehen ihm die Programmfunktionen „NetzwerkScan“, „Softwarelizenzen“ und „Auswertung“ zur Verfügung.

In der Funktion „NetzwerkScan“ kann der Benutzer die Softwareinventur im Netzwerk anstoßen. Hierfür benötigt der Benutzer jedoch Domain-Administrationsrechte. Während die Inventur läuft, werden eventuelle Fehler oder Ereignisse in einem Textfeld als „Scan – Log“ ausgegeben.

Programmintern laufen nun die Prozesse ab, die bereits unter 4.3 beschrieben wurden.



Programmfunktion – NetzwerkScan

In der Programmfunktion „Softwarelizenzen“ können die vorhandenen Rechnungen der jeweiligen Softwareprodukte erfasst werden. Hier werden die benötigten Informationen wie Produktname, Beschreibung, RechnungsNr., Rechnungsdatum, Einzelpreis, Gesamtpreis, Menge, Seriennummer und das Buchungskonto erfasst.

Die bereits erfassten Rechnungen werden in einem so genannten DBGrid angezeigt. Dies ist eine Art Datenbankfenster welches die ausgewählten Rechnungsinformationen der bereits erfassten Rechnungsdaten anzeigt.

GS - Inventur 1.0

Date ?

Netzwerkressourcen

New Group

NetzwerkScan

Softwarelizenzen

Auswertung

Softwarelizenzen :

Hinzufügen neuer Lizenzen :

Produktname : 1stClass Pro. for Delphi 3

Beschreibung :

Rechnungs NR.:

Rechnungs Datum : 12.09.2002

Einzelpreis : Gesamtpreis :

Menge : SN. NR.:

Konto :

Neu

bearbeiten

Hinzufügen

Vorhandene Lizenzen :

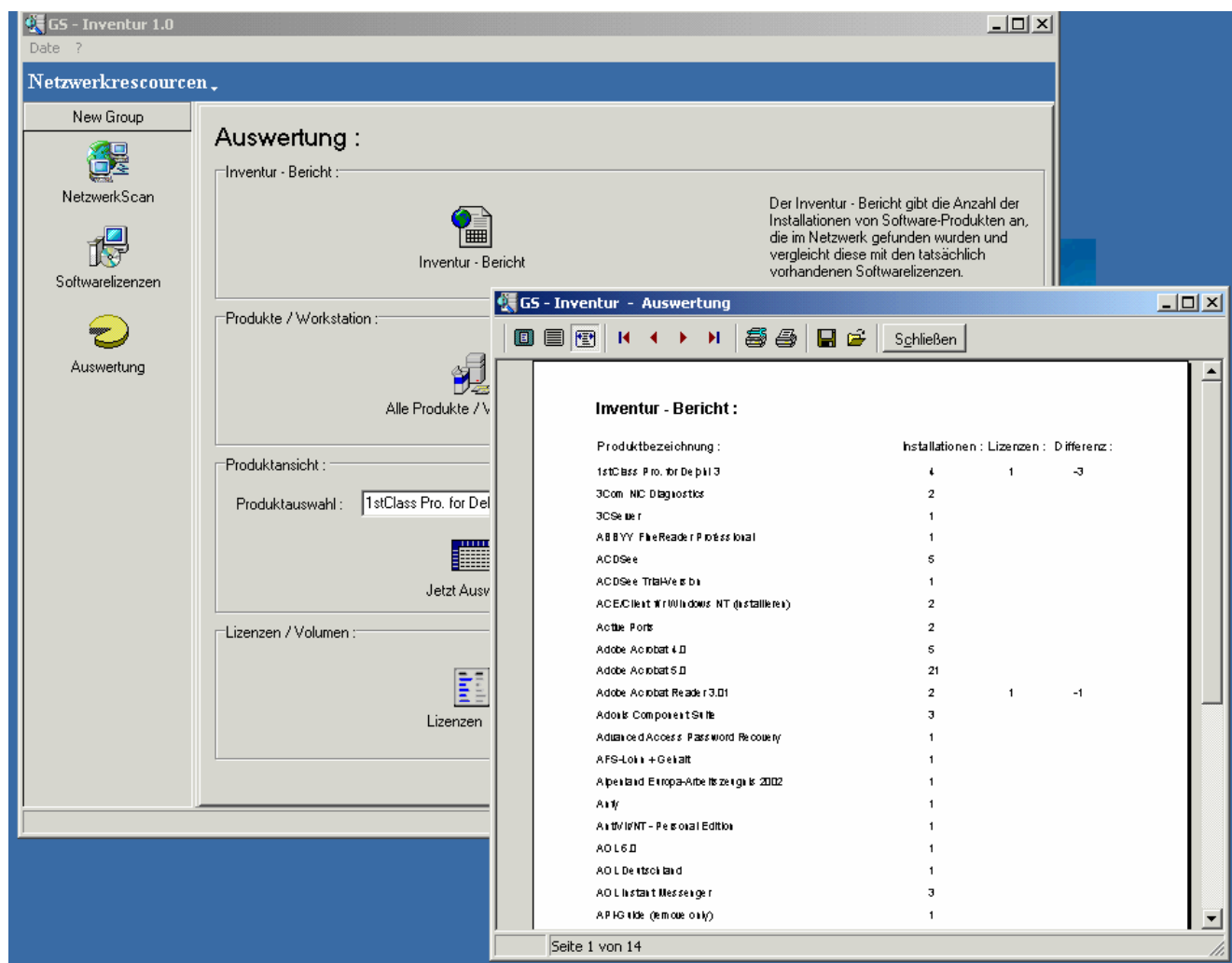
	ProduktName	Beschreibung	Rechnungs	Rechnungs	SN	Menge	Einzelpreis	Gesamtpreis	Konto
▶	1stClass Pro	1st Class Pro	ER 1692	30.06.1999		1	474,26	474,26	0480
	Adobe Acro	Adobe Acro	ER 385	26.03.1998	502976613	1	1164	1164	0480
	BDE Inform	Adobe Illust	ER 1119	28.05.1998		1	1245	1245	0400
	BDE Inform	Borland De	ER 778	10.06.1996		1	346,96	346,96	0480
	Borland De	Borland De		21.07.1997	HDB1330G	1	607,83	607,83	
	Borland De	Borland De	ER 1262	01.08.1997		4	781,74	3126,96	0480
	Borland De	Borland De	ER 2111	04.12.1997		1	42,6	42,6	0480
	Borland De	Borland De	ER 1975	05.08.1999		1	2585,34	2585,34	0027
	Borland De	Borland De	ER 1921	26.08.1998		1	688,79	688,79	0027
	Borland De	Borland De	ER 2691	27.09.1999		2	1981,9	1981,9	0027
	Borland De	Borland De		06.04.1995		1	450	450	0480

Programmfunktion – Softwarelizenzen

In der dritten Programmfunktion „Auswertung“, stehen dem Benutzer verschieden Auswertungsmöglichkeiten der Inventur zur Verfügung.

Hier gibt es zum einen den Inventurbericht, der alle installierte Software im Netzwerk anzeigt. Ebenso zeigt er die Anzahl der jeweiligen Softwareprodukte, die Anzahl der vorhandenen Lizenzen und die Differenz zwischen den vorhandenen Lizenzen und den realen Installationen an.

Es stehen noch drei weitere Auswertungsmöglichkeiten zur Verfügung, die dem Benutzer Aufschluss darüber geben welche Software auf welcher Workstation installiert ist, wie auch eine Auswertung über den Produktnamen, welche nur die Arbeitsstationen auswertet auf denen ein bestimmtes Produkt installiert ist. Ebenso eine Zusammenfassung der Lizenzvolumina, die Aufschluss über den Wert der im Unternehmen vorhandenen Lizenzen in Euro gibt.



Programmfunktion - Auswertung mit Ansicht eines Inventurberichts

Die grafische Benutzeroberfläche kann in ihrer Größe vom Benutzer individuell angepasst werden. Optimal ist sie jedoch für eine Bildschirmauflösung von 1024x768 Bildpunkten. Kleinere Auflösungen als 800x600 Bildpunkte sind nicht zu empfehlen, da dann die Ansicht der Anwendung unkomfortabel wird.

5 Abschließende Tests der Anwendung

Aus den Vorgaben des Pflichtenheftes ergaben sich folgende Testfälle:

- Den Programmablauf auf Korrektheit überprüfen, wie z.B.: Korrektes Datenbankhandling, Ausgabe/Anzeige der korrekten Daten aus der Datenbank.
- Erstellung der Auswertungen
- Drucken der Protokolle
- Daten der Arbeitsstationen in der Datenbank erfassen
- Überprüfen der Textbausteine der Benutzeroberfläche, Eingabefelder und Messageboxen auf grammatikalische Korrektheit und Verständnis.
- Testen der Validitätsprüfungen durch unlogische Eingaben an den Eingabefeldern. Handlung entgegen der Programmlogik.
- Die Programmfunktionen in verschiedenen Kombinationen und Einsatzfällen testen.

Anhand dieser Testszenarien können folgende Aussagen gemacht werden:

Zur Überprüfung der gewünschten Programmfunktionalität wurde ein Real-Test durchgeführt, indem ich die Anwendung von einem Domain-Administrator ausführen ließ. Hierbei wurde der Netzwerkscan ausgeführt. Dies ergab, dass der Netzwerkscan mehrere Minuten in Anspruch nimmt und von der Größe des Netzwerks abhängt, jedoch einwandfrei und stabil läuft. Die erkannten Workstations und die darauf installierte Software werden ordnungsgemäß in der Datenbank eingetragen.

Die Erstellung der Auswertung und das Drucken der Protokolle funktioniert ebenfalls einwandfrei. Es konnten keine Fehler festgestellt werden.

Zur Überprüfung der Textbausteine fügte ich diese in Microsoft Word ein und korrigierte mit Hilfe der Rechtschreibkorrektur einige Fehler.

Die Eingabefelder der Anwendung in der Programmfunktion Softwarelizenzen sind weitgehend frei von Validitätsprüfungen, da diese individuelle Eingabemöglichkeiten bieten müssen. Hier sind z.B. Felder dabei die sowohl Text als auch Zahlen aufnehmen müssen. Die Felder bei denen eine Fehleingabe vermieden werden muss sind z.B. durch eine Auswahl über eine „Combo Box“ oder wie beim Datum über einen „DatePicker“ von einer Fehleingabe des Benutzers befreit.

Ebenso habe ich die einzelnen SQL – Statements die in der Klasse „TGSDDataExchange“, für die Datenbankkommunikation oder in den Auswertungen zum Einsatz kommen, auf Fehler überprüft. Dazu habe ich die verschiedenen Statements z.B. daraufhin überprüft, ob auch leere Zeichenketten in die Datenbankfelder eingefügt werden können. Dies führte dazu, dass einige Felder der Datenbank in ihren Eigenschaften dahingehend geändert werden mussten, dass sie diese leeren Zeichenkette auch zulassen, sofern dies erlaubt war.

Abschließend lässt sich sagen, dass die Anwendung im Real – Test stabil und fehlerfrei lief. Ebenso zeigte sich, dass der Einsatz von ADO und Microsoft Access zu kurzen Zugriffszeiten von wenigen Sekunden auf die Datenbank führt und so geringe

Wartezeiten für den Benutzer entstehen. Die Datenbank wuchs beim ersten Real – Test schnell auf eine Größe von 1,9 MB an.

Nach Beendigung der Tests habe ich eine Setup - Routine mit dem Wise for Windows Installer erstellt. Diese installiert die Anwendung schnell und einfach auf einer gewünschten Arbeitsstation.

6 Fazit

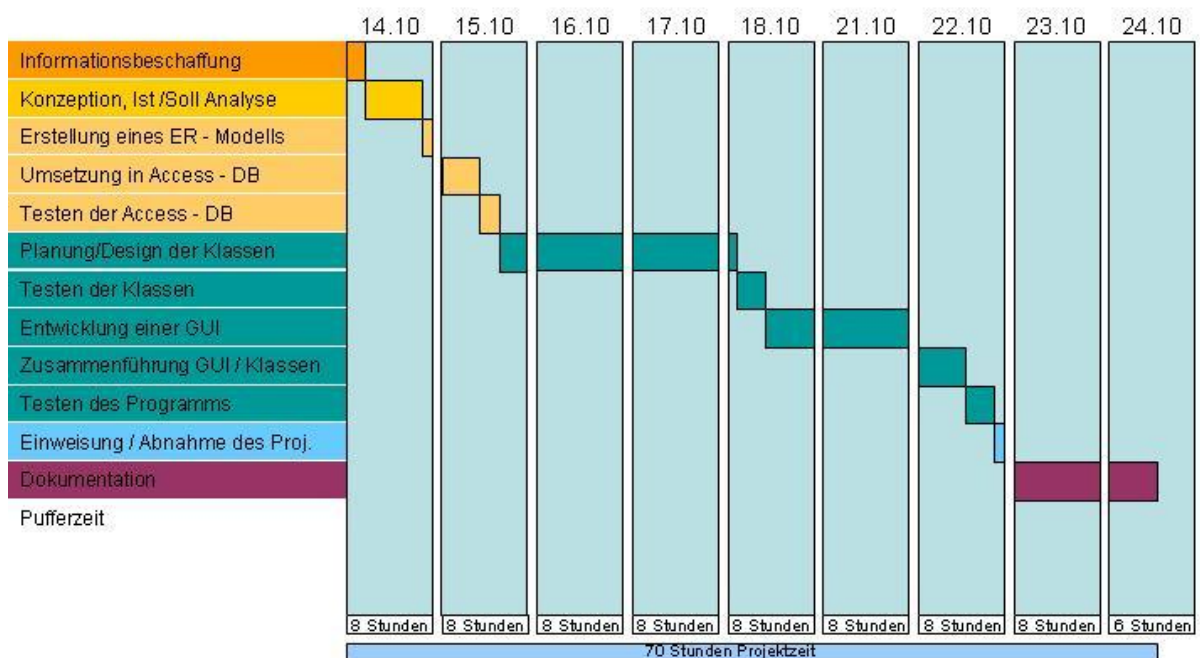
Abschließend lässt sich sagen, dass das Projekt anhand der Vorgaben des Pflichtenheftes und des Arbeitsauftrages der Geschäftsführung, sowie die Einweisung der Geschäftsführung erfolgreich durchgeführt wurde.

Die Projektziele und die im Pflichtenheft aufgeführten Muss-Kriterien wurden realisiert. Ebenso ist das Unternehmen durch den Einsatz der Software vor Übertretungen des UhrG (Uhrheberrechtsgesetz) geschützt.

Bei der Durchführung haben sich folgende Änderungen in der Zeitplanung ergeben:

In der Planungsphase „Konzeption Ist/Soll-Analyse“ konnte eine Stunde eingespart werden, die in der Phase „Planung/Design der Klassen“ wieder verwendet werden konnte. Die übrige Pufferzeit wurde für die Anfertigung der Dokumentation verwendet.

Projektphasenplan GS – Inventur



Durchführungszeitraum: 14.10 – 24.10.2002

Benjamin Wirtz

Die im Pflichtenheft aufgeführten Wunschkriterien werden zu einem späteren Zeitpunkt außerhalb der Projektzeit realisiert.

7 Anhang A Der Projektauftrag

04.07.2002

Innerbetrieblicher Projektauftrag

Projekt: Lizenz-Inventarverwaltung für die Gandke & Schubert GmbH & Co. KG

Projektverantwortlicher: Benjamin Wirtz

Projektzeitfenster: 70 Stunden im September/Oktober 2002 als Projektarbeit im Rahmen der Abschlussprüfung der Ausbildung zum Fachinformatiker

Ist-Zustand: Zur Zeit werden die PC sowie die Softwarelizenzen in der Buchhaltung erfasst und im Anlagevermögen separat verwaltet und ausgewiesen. Eine Zusammenführung von Rechnern und Lizenzen und die Inventur werden quartalsweise manuell vorgenommen.

Thema des Projekts: Erfassung und laufende gemeinsame Verwaltung der PC und der Software-Lizenzen in einer geeigneten zentralen Datenbank (zentrale Verwaltungseinheit ZVE).

Ziele des Projekts:

Es soll gewährleistet sein, dass die im Unternehmen eingesetzte Software immer nachweisbar lizenziert ist, um die Urheberrechtsbestimmungen einzuhalten

Es sollen Kostensparpotenziale realisiert werden, die durch die manuelle Verwaltung von PC und Lizenzen, durch die manuelle Inventur und manuell editierte Inventur- und Verschrottungsprotokolle anfallen

Der Nachweis der Lizenzierung soll zeitnah generiert werden können

Die ZVE soll über ein komfortabel zu bedienendes Front-end durch die Mitarbeiter/innen der Buchhaltung zu steuern sein

Muss-Kriterien:

Erstmaliger Abgleich der installierten und lizenzierten Software

Aufstellung ggf. erforderlichen Nachlizenzierungen

Eindeutige Zuordnung von Software-Lizenzen zu den PC

Verwaltungspool der erworbenen Lizenzen (abhängig vom Lizenzmodell)

Anzeige freier verfügbarer Lizenzen aus Lizenzpaketen

Automatische regelmäßige Inventur der Lizenzen durch ein Softwaretool, das in definierten Zeitintervallen die auf dem PC installierte Software an die ZVE meldet; automatischer Abgleich Ist-Soll mit Abweichungsprotokoll

Erstellen und Anbringen von Etiketten an den Rechnern, die zur Inventur durch ein Scannersystem erfasst werden und in die ZVE eingelesen werden kann

Ausgabe von Inventurprotokollen

Einweisung der Mitarbeiter/innen der Buchhaltung

Anwenderdokumentation

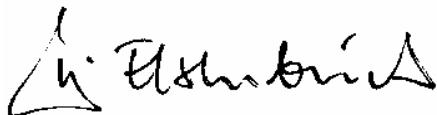
Wunsch-Kriterien

Schnittstelle zur Buchhaltungs-Software zur automatischen Übernahme eingebuchter PC bzw. Software-Lizenzen

Automatisches Generieren von Verschrottungsprotokollen bei Verschrottung von PC mit Reimportfunktion in die Anlagenbuchhaltung zum Ausbuchen des Anlageguts

Abgrenzungskriterien:

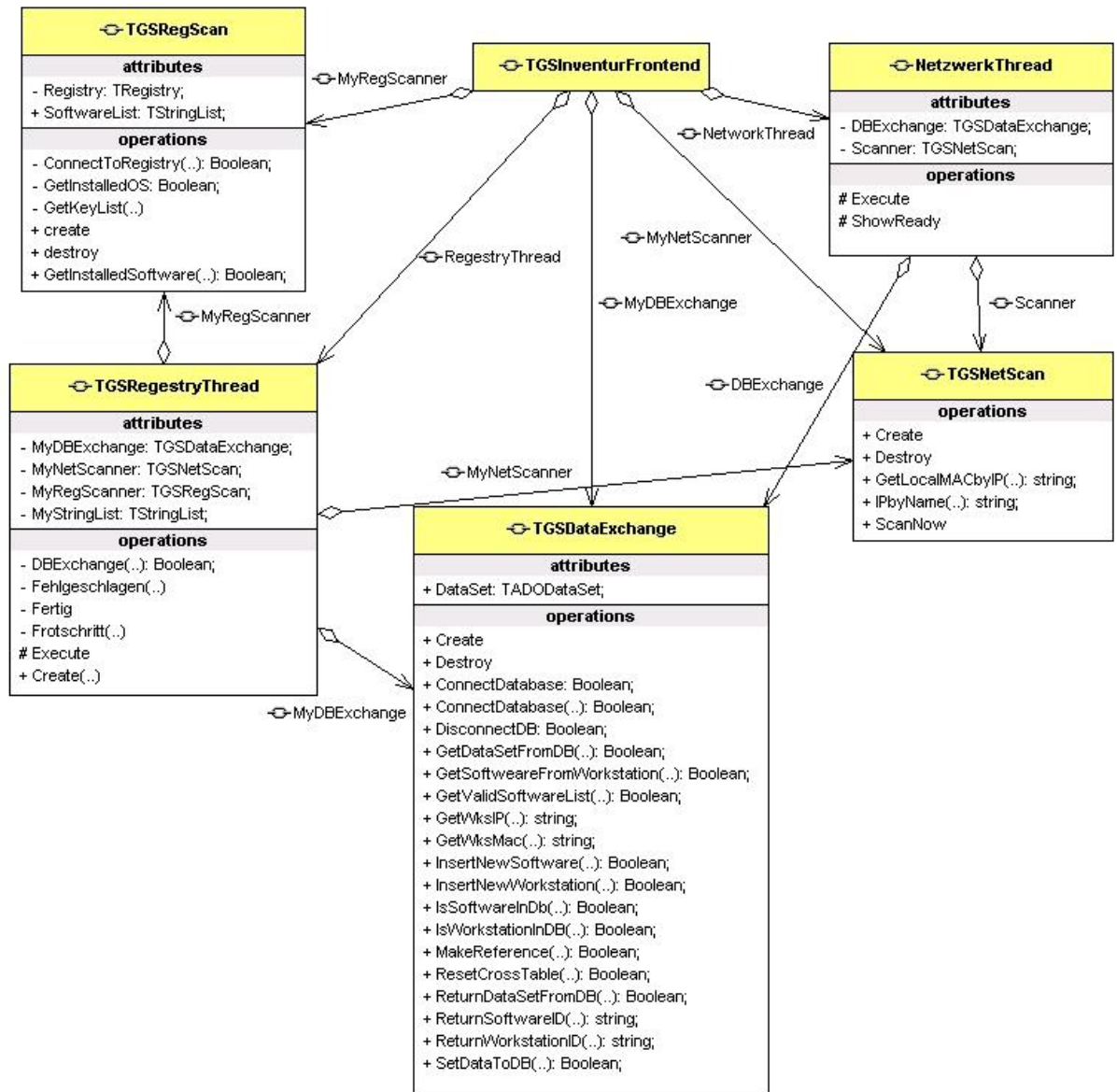
Die Erstellung von Benutzerprofilen (welcher Mitarbeiter benötigt welche Software) wird durch das IT-Management erstellt und ist nicht Projektbestandteil



Gandke & Schubert GmbH & Co. KG
Dipl.-Kfm. Frank Elsenbruch
Geschäftsführung

8 Anhang B

Das Klassendiagramm



Legende:

Vorzeichen der Methoden und Felder

+ = öffentliche Methode/Feld

- = private Methode/Feld

= geschützte (protected) Methode/Feld.

Die Pfeile zwischen den Klassen stellen Assoziationen („Hat“ – Referenzen) dar. Die Namen an den Assoziationen sind die Objektnamen der Klasseninstanzen.

9 Anhang C Der Sourcecode

Als Beispiel für den Sourcecode habe ich im folgenden den Quelltext der Klasse TGSDDataExchange aufgeführt.

```
{ ***** }
{      GS DataExchange2      }
{ Version: 1.0                }
{ Author:  Benjamin Wirtz    }
{ E-Mail:  bwirtz@gsn.de     }
{ Home Page: http://www.gsn.de }
{ Legal:   Copyright (c) 2002 by Gandke & Schubert GmbH & Co KG }
{ ***** }
```

```
unit UGS_DataExchange2;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
ExtCtrls, Db, ADODB, Grids, DBGrids, Registry;
```

```
type
```

```
TGSDDataExchange = class(TObject)
private
  fAdoConnection: TADOConnection;
  fAdoDataSet: TADODataSet;
  fAdoQuery: TADOQuery;
  function GetDataSet: TADODataSet;
  procedure SetDataSet(const Value: TADODataSet);
public
  constructor Create;
  destructor Destroy;
  function ConnectDatabase: Boolean; overload;
  function ConnectDatabase(const aConnectionString: String): Boolean; overload;
  function DisconnectDB: Boolean;
  function GetDataSetFromDB(const aSQLStatement: WideString): Boolean;
  function GetSoftwareFromWorkstation(WKSID: SString; var aAdoDataSet:
    TAdoDataSet): Boolean;
  function GetValidSoftwareList(aAdoDataSet: TADODataSet): Boolean;
  function GetWksIP(aID: Integer): string;
  function GetWksMac(aID: Integer): string;
  function InsertNewSoftware(aSoftwareName: String): Boolean;
  function InsertNewWorkstation(aIPAdress: String; aMacAdress: String;
    aClientName: String): Boolean;
  function IsSoftwareInDb(aSoftwareName: String): Boolean;
  function IsWorkstationInDB(aIPAdress: String): Boolean;
  function MakeReference(aIPAdress: String; aSoftwareList: TStringList):
    Boolean;
  function ResetCrossTable(WKSID: SString): Boolean;
  function ReturnDataSetFromDB(const aSQLStatement: WideString; var
    aAdoDataSet: TAdoDataSet): Boolean;
  function ReturnSoftwareID(aSoftName: SString): string;
  function ReturnWorkstationID(aIPAdress: SString): string;
  function SetDataToDB(const aSQLStatement: WideString): Boolean;
  property DataSet: TADODataSet read GetDataSet write SetDataSet;
```

end;

implementation

```
{
***** TGSDataExchange *****
}
constructor TGSDataExchange.Create;
begin
    inherited Create;
    fAdoDataSet:= TADODataset.Create(nil);
    fAdoConnection:= TADOConnection.Create(nil);
    fAdoQuery:=TADOQuery.Create(nil);
end;

destructor TGSDataExchange.Destroy;
begin
    if fAdoConnection.Connected then
        DisconnectDB;
    freeandnil(fAdoConnection);
    if assigned(fAdoDataSet) then
        freeandnil(fAdoDataSet);
    freeandnil(fAdoQuery);

    inherited Destroy;
end;

function TGSDataExchange.ConnectDatabase: Boolean;
var
    MyRegistry: TRegistry;
    MyConnectionString: string;
    sPath: string;

const
    PathKey='\SOFTWARE\GS\INVENTUR';
    DbPath='DBPath';
    DbProvider='DBProvider';

begin
    try
        MyRegistry:=TRegistry.Create;
        MyRegistry.RootKey:=HKEY_LOCAL_MACHINE;
        MyRegistry.CreateKey(PathKey);
        MyRegistry.OpenKey(PathKey,TRUE);
        sPath:= Extractfilepath((Application.ExeName))+ 'Lizenz.mdb';
        MyRegistry.WriteString(DbPath, sPath);
        MyRegistry.WriteString(DbProvider, 'Microsoft.Jet.OLEDB.4.0');
        MyConnectionString:='Provider='+MyRegistry.ReadString(DbProvider)+';';
        MyConnectionString:=MyConnectionString+'Data Source='+MyRegistry.ReadString(DbPath)+';';
        result:=ConnectDatabase(MyConnectionString);
    finally
        freeandnil(MyRegistry);
    end;
end;

function TGSDataExchange.ConnectDatabase(const aConnectionString: String): Boolean;
begin
    result := True;
    fAdoConnection.ConnectionString:=aConnectionString;
    fAdoConnection.LoginPrompt:=False;
    Try
```

```

    fAdoConnection.Connected:=True;
except
    freeandnil(fAdoConnection);
    freeandnil(fAdoDataSet);
    freeandnil(fAdoQuery);
    result:=false;
    Exit;
End;
end;

function TGSDDataExchange.DisconnectDB: Boolean;
begin
    result:=true;
    try
        fAdoConnection.Connected:=False;
        fAdoConnection.Close;
    except
        result:=false;
    end;
end;

function TGSDDataExchange.GetDataSet: TADODataset;
begin
    if not assigned(fAdoDataSet) then
        fAdoDataSet:=TADODataset.Create(nil);
    result:=fAdoDataSet;
end;

function TGSDDataExchange.GetDataSetFromDB(const aSQLStatement: WideString):
    Boolean;
begin
    result:=ReturnDataSetFromDB(aSQLStatement,fadodataset);
end;

function TGSDDataExchange.GetSoftwareFromWorkstation(WKSID: SString;var
    aAdoDataSet: TAdoDataSet): Boolean;
var
    SQL: string;
begin
    try
        SQL:='SELECT ProduktName FROM TWorkstation INNER JOIN (TInstalledSoftware INNER JOIN
TIsInstalledOn ON TInstalledSoftware.TSoftwareID=TIsInstalledOn.TInstalledSoftwareID)';
        SQL:=SQL+' ON TWorkstation.TWorkstationID=TIsInstalledOn.TWorkstationID WHERE
TWorkstation.TWorkstationID='+WKSID+'';
        if ReturnDataSetFromDB(SQL,aAdoDataSet) then result:=TRUE;
    except
        result:=FALSE;
    end;
end;

function TGSDDataExchange.GetValidSoftwareList(aAdoDataSet: TADODataset):
    Boolean;
var
    SQL: string;
begin
    SQL:='SELECT * FROM TInstalledSoftware ORDER BY TInstalledSoftware.ProduktName;';
    result:=False;
    try
        ReturnDataSetFromDB(SQL,aAdoDataSet);
    except
        result:=False;
    exit;
end;

```

```

end;
result:=TRUE;
end;

function TGSDataExchange.GetWksIP(aID: Integer): string;
begin
  GetDataSetFromDB('SELECT TWorkstation.IPAдресse FROM TWorkstation WHERE
TWorkstation.TWorkstationID='+IntToStr(aID)+'');
  if fAdoDataSet.Recordset.RecordCount > 0 Then
  begin
    fAdoDataSet.open;
    fAdoDataSet.first;
    result:=fAdoDataSet.fieldbyname('IPAdresse').AsString;
    fAdoDataSet.close;
  end;
end;

function TGSDataExchange.GetWksMac(aID: Integer): string;
begin
  GetDataSetFromDB('SELECT TWorkstation.MACAdresse FROM TWorkstation WHERE
TWorkstation.TWorkstationID='+IntToStr(aID)+'');
  if fAdoDataSet.Recordset.RecordCount > 0 Then
  begin
    fAdoDataSet.open;
    fAdoDataSet.first;
    result:=fAdoDataSet.fieldbyname('MACAdresse').AsString;
    fAdoDataSet.close;
  end;
end;

function TGSDataExchange.InsertNewSoftware(aSoftwareName: String): Boolean;
begin
  fAdoConnection.BeginTrans;
  try
    SetDataToDB('INSERT INTO TInstalledSoftware (ProduktName) VALUES ('+ QuotedStr(aSoftwareName)
+')');
  except
    fAdoConnection.RollbackTrans;
    result:=False;
    exit;
  end;
  fAdoConnection.CommitTrans;
  result:=True;
end;

function TGSDataExchange.InsertNewWorkstation(aIPAdress: String;
  aMacAdress:String;aClientName: String): Boolean;
begin
  fAdoConnection.BeginTrans;
  try
    SetDataToDB('INSERT INTO TWorkstation (IPAdresse, MACAdresse, ClientName) VALUES ('+
QuotedStr(aIPAdress) +', '+ QuotedStr(aMacAdress) +', '+ QuotedStr(aClientName) +')');
  except
    fAdoConnection.RollbackTrans;
    result:=False;
    exit;
  end;
  fAdoConnection.CommitTrans;
  result:=True;
end;

function TGSDataExchange.IsSoftwareInDb(aSoftwareName: String): Boolean;

```

```

begin
  result:=False;
  try
    GetDataSetFromDB('SELECT ProduktName FROM TInstalledSoftware WHERE(ProduktName = '+
QuotedStr(aSoftwareName) + ');');
    if fAdoDataSet.Recordset.RecordCount > 0 then
      begin
        result:=True;
      end;
    except
      result:=False;
    end;
  end;

function TGSDDataExchange.IsWorkstationInDB(aIPAdress: String): Boolean;
begin
  result:=False;
  try
    //if not Assigned(fAdoConnection) then ConnectDatabase;
    GetDataSetFromDB('SELECT IPAdresse FROM TWorkstation WHERE (IPAdresse = ' +
QuotedStr(aIPAdress) + ');');
    if fAdoDataSet.Recordset.RecordCount > 0 then result:=True;
  except
    result:=False;
    exit;
  end;
end;

function TGSDDataExchange.MakeReference(aIpAdress: String;aSoftwareList:
  TStringList): Boolean;
var
  i: Integer;
  WSID: string;
  SoftID: TStringList;
begin
  SoftID:=TStringList.Create;
  result:=False;
  fAdoConnection.BeginTrans;
  try
    //WorkstationID wird ermittelt
    WSID:= ReturnWorkstationID(aIpAdress);
    if WSID = '' then
      begin
        showmessage('Fehler WSID');
        Result:=False;
        exit;
      end;
    for i:=0 to aSoftwareList.Count-1 do
      begin
        SoftID.Add(ReturnSoftwareID(aSoftwareList.Strings[i]));
      end;
    ResetCrossTable(WSID);
    for i:=0 to SoftID.Count-1 do
      begin
        SetDataToDB('INSERT INTO TIsInstalledOn (TWorkstationID, TInstalledSoftwareID) VALUES ('+
QuotedStr(WSID) +', '+ QuotedStr(SoftID.Strings[i] +')');
      end;
    except
      fAdoConnection.RollbackTrans;
      result:=False;
      exit;
    end;
  end;
end;

```

```

fAdoConnection.CommitTrans;
result:=True;
end;

function TGSDDataExchange.ResetCrossTable(WKSID: SString): Boolean;
begin
result:=True;
try
if SetDataToDB('DELETE * FROM TIsInstalledOn WHERE(TWorkstationID = '+ WKSID+');') then
result:=TRUE
else result:=False;
except
result:=False;
exit;
end;
end;

function TGSDDataExchange.ReturnDataSetFromDB(const aSQLStatement: WideString;
var aAdoDataSet: TAdoDataSet): Boolean;
begin
try
with aAdoDataSet do begin
Connection:=fAdoConnection;
Close;
CommandText:=aSQLStatement;
Open;
if aAdoDataSet.Recordset.RecordCount > 0 then result:=TRUE;
end;
except
result:=FALSE;
end;
aAdoDataSet.Connection:=nil;
end;

function TGSDDataExchange.ReturnSoftwareID(aSoftName: SString): string;
var
SoftID: Integer;
begin
//// Software ID wird festgestellt.
GetDataSetFromDB('SELECT TSoftwareID FROM TInstalledSoftware WHERE(ProduktName = '+
QuotedStr(aSoftName) + ');');
if fAdoDataSet.Recordset.RecordCount > 1 then
begin
result:="";
exit;
end
else begin
SoftID:=fAdoDataSet.fieldbyname('TSoftwareID').AsInteger;
result:=IntToStr(SoftID);
end;
end;

function TGSDDataExchange.ReturnWorkstationID(aIpAddress: SString): string;
var
WSID: Integer;
begin
//// WORKSTATION ID wird festgestellt.
if not assigned(fAdoDataSet) then
fAdoDataSet:=TADODataset.Create(nil);

GetDataSetFromDB('SELECT TWorkstationID FROM TWorkstation WHERE(IPAdresse = '+
QuotedStr(aIpAddress) + ');');

```

```

if fAdoDataSet.Recordset.RecordCount > 1 then
begin
    result:="";
    exit;
end
else begin
    WSID:=fAdoDataSet.fieldbyname('TWorkstationID').AsInteger;
    result:=IntToStr(WSID);
end;
end;

procedure TGSDDataExchange.SetDataSet(const Value: TADODataset);
begin
    ShowMessage('Falscher Weg !');
end;

function TGSDDataExchange.SetDataToDB(const aSQLStatement: WideString): Boolean;
begin
    result:=true;
    try
        with fAdoQuery do begin
            fAdoQuery.Connection:=fAdoConnection;
            Close;
            with SQL do begin
                Clear;
                Add(aSQLStatement);
            end;
            ExecSQL;
        end;
    except
        result:=false;
        exit;
    end;
end;

end.

```

P f l i c h t e n h e f t

Anwendung: GS-Inventur

Entwickler: Benjamin Wirtz

Datum: 13.09.2002

Quelle: "Helmut Balzert, Lehrbuch der
Software-Technik"
ISBN-Nr. 3827403588

INHALTSVERZEICHNIS

1	REVISIONEN.....	32
1.1	Übersicht der Revisionshistorie	32
2	ZIELBESTIMMUNGEN.....	33
2.1	Muss-Kriterien	33
2.2	Wunsch-Kriterien	33
2.3	Abgrenzungs-Kriterien.....	34
3	PRODUKTEINSATZ	34
3.1	Anwendungsbereich.....	34
3.2	Zielgruppe.....	34
4	PRODUKTUMGEBUNG	34
4.1	Software	34
4.2	Hardware	34
5	PRODUKTFUNKTIONEN.....	35
5.1	Funktion 1 – Erfassung der Daten einer Arbeitsstation.	35
5.2	Funktion 2 – Verwalten der Lizenzdaten.....	35
5.3	Funktion 3 – Auswertungen	35
6	PRODUKTDATEN	35
6.1	Produktdaten 1 – Arbeitsstationsdaten.....	35
6.2	Produktdaten 2 – Lizenzdaten	35
7	QUALITÄTSBESTIMMUNGEN.....	36
7.1	Abbildung in Tabellarischer Form.....	36
8	BENUTZERSCHNITTSTELLEN	36
9	GLOBALE TESTFÄLLE.....	36
10	ENTWICKLUNGSUMGEBUNG	37
10.1	Software	37
10.2	Hardware	37

1 Revisionen

1.1 Übersicht der Revisionshistorie

Unter diesem Punkt werden nachträgliche Änderungen am Dokument selbst festgehalten, wobei immer nur die Änderungen von einer Revision zur nächsten unter einem Punkt zusammen mit dem Änderungsdatum festgehalten werden. Die Uhrversion des Dokumentes hat die Versionsnummer 1.0.

1. 13.09.2002 Version 1.0 / keine Änderung.

2 Zielbestimmungen

In der Gandke & Schubert GmbH & Co KG werden die verschiedensten Softwareprodukte auf verschiedenen Arbeitsstationen für die Bewältigung der täglichen Arbeit eingesetzt. Für diese Softwareprodukte müssen immer ausreichend Softwarelizenzen vorhanden sein, um das Unternehmen vor der Übertretung des UrhG (Urheberrechtsgesetz) zu schützen.

Es soll ein Delphi5-Programm entwickelt werden, welches gewährleistet, dass die im Unternehmen eingesetzte Software immer nachweisbar lizenziert ist. Durch diese Software sollen Kostensparpotenziale realisiert werden, die durch die manuelle Verwaltung von PC und Lizenzen, durch die manuelle Inventur und manuell editierten Inventur- und Verschrottungsprotokolle anfallen.

Die Daten der PC und Softwarelizenzen soll in einer zentralen Datenbank verwaltet werden und es sollen zeitnahe Auswertungen und Protokolle erstellt werden können.

2.1 Muss-Kriterien

- Erstellung eines erstmaligen Abgleichs aller installierten und lizenzierten Software im Hause Gandke & Schubert und Erstellung einer Aufstellung für evtl. erforderliche Nachlizenzierung.
- Es muss eine eindeutige Zuordnung der Softwarelizenzen zu den einzelnen PCs im gesamten Netzwerk möglich sein und die Möglichkeit einer manuellen Anpassung bestehen.
- Die einzelnen Lizenzen sollen in einem Verwaltungspool aufgeführt werden. Außerdem soll hier angezeigt werden aus welchem Lizenzpaket noch freie Lizenzen zur Verfügung stehen.
- Eine automatische, zeitnahe Inventur soll in regelmäßigen Zeitintervallen durchgeführt werden und alle auf den PCs installierte Software an eine Zentrale Datenbank melden. Hierbei soll ein automatischer Ist/Soll – Abgleich mit Abweichungsprotokoll erstellt werden.
- Ausgabe aller erzeugten Inventurprotokolle.
- Einweisung des Geschäftsführers in die Software.

2.2 Wunsch-Kriterien

- Implementierung einer Schnittstelle zur Buchhaltungs- – Software zur automatischen Übernahme eingebuchter PC bzw. Software – Lizenzen.
- Eine automatische Generierung von Verschrottungsprotokollen bei Verschrottung von PC mit Reimportfunktion in die Anlagenbuchhaltung zum Ausbuchen des Anlageguts.
- Erstellen und Anbringen von Etiketten an den Rechnern, die zur Inventur durch ein Scannersystem erfasst werden und in die Software eingelesen werden.

2.3 Abgrenzungs-Kriterien

- Die Erstellung von Benutzerprofilen (welcher Mitarbeiter benötigt welche Software) wird durch das IT-Management erstellt und ist nicht Projektbestandteil

3 Produkteinsatz

3.1 Anwendungsbereich

Die Anwendung soll die Daten der auf den einzelnen Arbeitsstationen installierten Software in der Firma Verwalten. Es soll zu erkennen sein welche Software ordnungsgemäß Lizenziert ist und welche ggf. Nachlizenziert werden muss. Somit soll die Anwendung den Mitarbeiter/innen der Buchhaltung die Inventur, Verwaltung des Ist/Soll – Abgleichs erleichtern.

Von hier können die Auswertungen angestoßen werden und Ergebnisse und Auswertungen überwacht werden.

Zur Dokumentierung können die Inventurprotokolle ausgedruckt werden.

3.2 Zielgruppe

Die Geschäftsführung.

4 Produktumgebung

4.1 Software

Das Programm wird unter Borland Delphi 5 (Version Enterprise) und Microsoft Access 2002 entwickelt und ist unter Windows 2000, XP lauffähig.

Zur späteren Verwendung wird die Access – Datenbank auf den im Hause Gandke & Schubert vorhandenen Microsoft SQL – Server 2000 portiert, da dieser als zentrales RDBMS eingesetzt wird.

4.2 Hardware

Die Anwendung setzt als Mindestanforderung einen IBM kompatiblen PC mit mindestens 233 Megahertz CPU, 64 Megabyte Arbeitsspeicher, einer 100MBit Netzwerkkarte und eine VGA kompatible Grafikkarte mit einer Mindestauflösung von 800 * 600 Bildpunkten voraus.

5 Produktfunktionen

5.1 Funktion 1 – Erfassung der Daten einer Arbeitsstation.

- F10 / Erfassung aller installierten Software einer Arbeitsstation.
- F20 / Erfassung von Hardwaremerkmalen.
- F30 / Zentrale Verwaltung der Arbeitsstationsdaten.
- F40 / Manuelles hinzufügen, löschen, ändern der Arbeitsstationsdaten.

5.2 Funktion 2 – Verwalten der Lizenzdaten

- F50 / Zentrale Erfassung und Verwaltung von Softwarelizenzen.
- F60 / Zuordnung der Softwarelizenzen zu den einzelnen Arbeitsstationen.
- F70 / Manuelles hinzufügen, löschen, ändern der Lizenzdaten.

5.3 Funktion 3 – Auswertungen

- F80 / Ist/Soll – Abgleich der verwendeten Softwarelizenzen in definierten Zeitabständen.
- F90 / Erstellung von Auswertungsgrafiken und Inventurprotokollen.
- F100 / Ausdrucken der Inventurprotokolle.

6 Produktdaten

6.1 Produktdaten 1 – Arbeitsstationsdaten

Von den einzelnen Arbeitsstationen sind folgende Daten zu erfassen:

- D10 / MAC – Adresse, IP – Adresse und Rechnername zur Identifikation der Arbeitsstation.
- D20 / Name und Version der installierten Softwareprodukte.

6.2 Produktdaten 2 – Lizenzdaten

Über die einzelnen Lizenzpakete sind folgende Daten zu speichern:

- D30 / Produktname der jeweiligen Lizenz.
- D40 / Anzahl der Lizenzen für jedes Produkt.
- D50 / Wird die jeweilige Lizenz beansprucht und wie oft.
- D60 / Beziehung zur jeweiligen Arbeitsstation.
- D70 / Differenzen des Ist/Soll - Abgleichs.

7 Qualitätsbestimmungen

7.1 Abbildung in Tabellarischer Form

Produktqualität	Sehr gut	gut	Normal	Nicht relevant
Funktionalität		X		
Komfort	X			
Wirtschaftlichkeit			X	
Sicherheit	X			
Dokumentation				
Vollständigkeit	X			
Verständlichkeit		X		
Änderbarkeit		X		
Modifizierbarkeit		X		
Stabilität		X		

8 Benutzerschnittstellen

Die Benutzerschnittstelle ist für Tastatur – sowie Mausbenutzung auszulegen.

9 Globale Testfälle

Die Tests zu den oben angegebenen Leistungsanforderungen werden durch den Programmierer vorgenommen. Getestet werden folgende Punkte:

- Den Programmablauf auf Korrektheit überprüfen, wie z.B.: Korrektes Datenbankhandling, Ausgabe/Anzeige der korrekten Daten aus der Datenbank.
- Erstellung der Auswertungen
- Drucken der Protokolle
- Daten der Arbeitsstationen in der Datenbank erfassen
- Lizenzen zuweisen.
- Überprüfen der Textbausteine der Benutzeroberfläche, Eingabefelder und Messageboxen auf grammatikalische Korrektheit und Verständnis.
- Testen der Validitätsprüfungen durch unlogische Eingaben an den Eingabefeldern. Handlung entgegen der Programmlogik.
- Die Programmfunktionen in verschiedenen Kombinationen und Einsatzfällen testen.

10 Entwicklungsumgebung

10.1 Software

Das Programm wird auf Basis der Delphi 5 Enterprise Entwicklungsumgebung und MS – Access entwickelt. Das Betriebssystem ist Windows 2000.

Die benötigten Klassen werden den jeweiligen Problemstellungen entsprechend entwickelt. Zur Erstellung der Benutzerschnittstelle werden die Klassen der Delphi 5 VCL (Virtuelle Klassenbibliothek) und die Firmeneigenen Klassen verwendet.

10.2 Hardware

Der Entwicklungsrechner ist ein Pentium II mit 266 Mhz und 196 MB Ram.