Netzwerkdiagnose in einer Produktionsumgebung

Erstellung eines Visualisierungsmodells zur Netzwerkdiagnose
in einer automatisierten Produktionsumgebung

Diplomarbeit

angefertigt an der
Universität Lüneburg,
Fachbereich Automatisierungstechnik,
Studiengang Angewandte Automatisierungstechnik
zur Erlangung des akademischen Grades eines
Dipl.-Ing. (FH)

Vorgelegt von: Armin Anjarwalla (Matrikelnummer 1155236)

Erstprüfer: Prof. Dr.-Ing. Philipp Odensass

Zweitprüfer: Prof. Dr.-Ing. Klaus-Dieter Hübner

Abgabedatum: 06.03.2007

Vorwort

Während meiner Diplomarbeit, die ich bei der Firma Rockwell Automation erstellt habe, hatte ich die Möglichkeit, über das Thema dieser Arbeit hinaus Erfahrungen zu sammeln. Mir wurde die Möglichkeit geboten, auch an Projekten für andere Kunden mitzuarbeiten, wodurch ich mir einen Eindruck der Vielfältigkeit dieser Arbeit machen konnte. Dadurch haben sich immer wieder neue Ideen für diese Arbeit ergeben. Das Thema dieser Arbeit ergab sich aus dem Wunsch, einen zentralen Zugang zu Informationen über den Status einer Visualisierungsapplikation zu bekommen, um dadurch eine gezielte und strukturierte Analyse zu ermöglichen.

Während des ersten und zweiten Praxissemesters wurde ich in der Firma von Herrn Anthony Cargan betreut. Für die Unterstützung und Förderung möchte ich mich an dieser Stelle ganz herzlich bei ihm bedanken.

Mein Dank geht auch an Herrn Prof. Dr.-Ing. Philipp Odensass für die Betreuung während des ersten und zweiten Praxissemesters.

Lüneburg, im März 2007

Armin Anjarwalla

Inhaltsverzeichnis

1 Einleitung

Der Einsatz und die Bedeutung von Visualisierungssystemen nehmen in einer modernen Produktionsumgebung immer weiter zu. Ein System, dass neue Maßstäbe gesetzt hat, ist die eingesetzte Anlagenvisualisierung von DaimlerChrysler im Werk Düsseldorf. An diesem Standort wird seit dem Jahr 2006 unter dem Projektnamen „NCV3“ die aktuelle Sprinterfamilie produziert. Das Werk gehört seit 1962 zur Daimler-Benz AG aus der, durch Fusion mit der Chrysler Corporation, die DaimlerChrysler AG entstand.

Die Produktion erfolgt auf einer einzigen Produktionslinie auf der ca. 500 Fahrzeuge pro Tag gefertigt werden. Dieser Aufbau der Produktion bietet im Gegensatz zu einer Produktion mit parallelen Produktionslinien den Vorteil, dass weiniger Fläche und weniger Bedienpersonal benötigt wird. Durch dieses Konzept werden die Anforderungen an Verfügbarkeit und Bedienbarkeit der Visualisierungsanwendung sowie die Ansprüche an die Leittechnik deutlich erhöht, da ein Ausfall von Produktionsanlagen sehr starke Auswirkungen auf die gesamte Anlage hat. In dieser auftragsbezogenen Produktion können vom Kunden über 64000 Karosserievarianten bestellt werden so das zu beobachten ist, dass sich fast jede gefertigte Karosserie auf diese Produktionslinie von der vorherigen unterscheidet.

Als Visualisierungssystem wird in dem Bereich „Rohbau“ die Visualisierungssoftware „RSView SE“ der Firma Rockwell Automation eingesetzt. Rockwell Automation ist ein weltweit tätiges Unternehmen, das seinen Kunden in mehr als 80 Ländern industrielle Produkte und Dienstleistungen in den Bereichen Antriebs-, Steuerungs- und Informationstechnologien anbietet. Die Visualisierungssoftware von Rockwell Automation ermöglicht den Betrieb in drei Varianten der Visualisierung. Es stehen die Produkttypen „Machine Edition“, „Stand Alone“ und „Distributed“ zur Verfügung. Die ersten beiden Varianten eigenen sich zum Einsatz an kleineren Anlagen, die ein Bedienterminal erfordern. Beim Einsatz der „Machine Edition“ kann die Visualisierung sogar auf einem Windows CE basierenden Gerät laufen. Die Größe der Produktionsanlage des Sprinterrohbaus erfordert jedoch eine, in der ganzen

Produktion verteilte, Visualisierung. Hier kommt eine „Distributed", also verteilte oder dezentrale Visualisierung zum Einsatz.

Durch den Einsatz von RSView SE können die Aufgaben eines „Human Machine Interface" Systems auf verschiedenen Ebenen gelöst werden. Im maschinennahen Bereich stehen Werkerbedienstationen zur Verfügung, die Leittechnik kann das System als übergreifendes Visualisierungstool nutzen. Diese Ausdehnung der Produktionsanlage setzt besondere Ansprüche an die Visualisierung. So werden inzwischen über 250 „Full HMI Clients" eingesetzt auf denen keine eigenständige Visualisierungsapplikation installiert ist. Die Clients greifen auf mehrere Server zu, die alle benötigten Bilder und Daten bereitstellen. Durch diese Aufteilung ist eine der komplexesten industriellen Visualisierungsapplikationen überhaupt entstanden.

2 Techniken

Während der Erstellung des zur Diplomarbeit gehörenden Projektes wurden verschiedenste Techniken eingesetzt. Die wesentlichen und aus der eigentlichen Entwicklung des Projektes losgelösten Techniken werden im Folgenden vorgestellt.

2.1 RSView SE

RSView SE ist das Visualisierungsprogramm von Rockwell Software. Mit diesem Programm ist es möglich, Maschinen und ganze Anlagen zu überwachen und zu steuern. Insgesamt stehen drei verschiedene Varianten von Visualisierungsapplikationen zur Verfügung *[2.1]*.

Mit einer RSView ME (Machine Edition) Anwendung kann eine Visualisierung für Windows CE Geräte erstellt werden. Hierbei werden im Prinzip die drei benötigten Komponenten (HMI Server zum Bereitstellen der Visualisierungsbilder, Dataserver um die Verbindung zur Steuerung herzustellen und Client zur Anzeige der Daten) nach dem Erstellen der Anwendung zu einem speziellen Programm „verpackt". Nach diesem „Verpacken" kann die „ME Runtime", also das Programm für den normalen Betrieb, auf einen entsprechenden Client runtergeladen werden.

Die zweite Variante erlaubt die Nutzung eines normalen PCs mit Windows Betriebssystem. Hier befinden sich die einzelnen Komponenten, ähnlich wie bei einer ME Applikation, ebenfalls alle auf einem PC, brauchen aber nicht „verpackt" zu werden. In solch einer Visualisierungsapplikation stehen mehr Darstellungsmöglichkeiten zur Verfügung als bei einer ME Applikation (was zum Teil schon durch bessere Bedien- und Anzeigemöglichkeiten eines PCs erklärt wird). Eine solche Applikation heißt entsprechend der Einsatzmöglichkeiten „Stand-Alone" Applikation.

Die dritte Variante von RSView SE Applikationen ist eine Produktvariante speziell für verteilte Visualisierungen. Das bedeutet, dass einzelne Funktionen auf

mehrere Server aufgeteilt werden können. Eine solche Applikation wird „Distributed“ Applikation genannt.

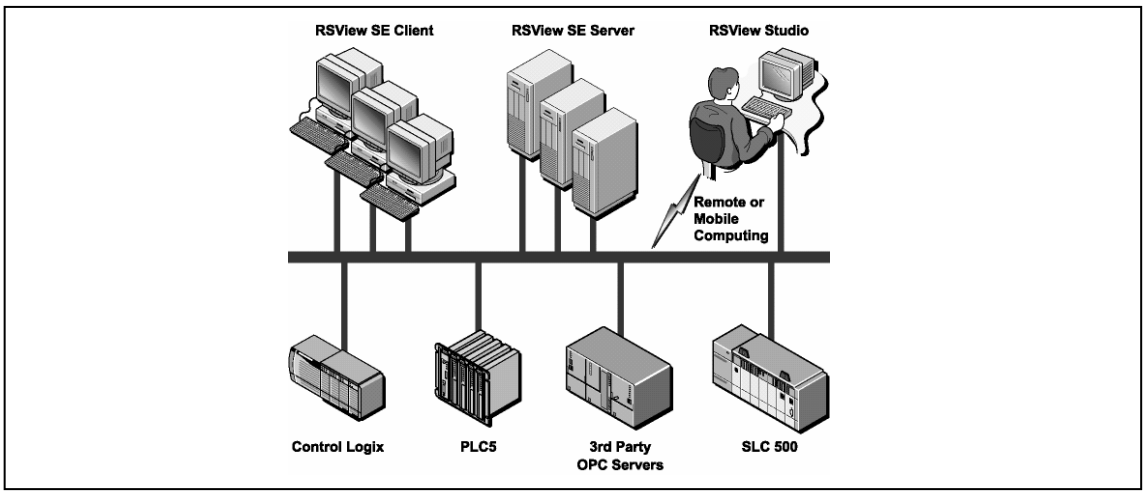


Abb. 1: Verteilte Visualisierungsarchitektur

Diese Verteilung der Visualisierung ermöglicht nicht nur eine höhere Leistungsfähigkeit, sondern stellt auch besondere Ansprüche an die Umgebung in der die Applikation läuft. So wird ein Dienst benötigt, in dem alle Komponenten aufgelistet sind, und es wird empfohlen, die Applikation in einer Windowsdomäne zu betreiben.

2.1.1 Komponenten einer verteilten RSView SE Applikation

Nach der Fertigstellung einer Anwendung sind vor allem die Server der Applikation von Bedeutung, die verschiedene Dienste anbieten. Darüber hinaus werden aber auch Clients verwendet, und es existiert eine Entwicklungsumgebung zum Erstellen der Applikationen.

2.1.1.1 HMI Server

Der HMI Server stellt vor allem die auf diesem Server gespeicherten Visualisierungsbilder zur Verfügung. Des Weiteren kann dieser Server zum Sammeln von Daten (Datalogging), Alarmbehandlung und historischer Auswertung von Daten genutzt werden. Ein Server ist in diesem Fall als logische Einheit innerhalb des RSView SE Projektes zu verstehen. Ein HMI Server kann aus bis zu zwei physikalischen Computern bestehen. Andererseits können auf einem physikalischen Computer mehrere HMI Server installiert sein. Dieses Vorgehen ist

allerdings für den produktiven Einsatz nicht empfohlen, kann jedoch zu Entwicklungszwecken genutzt werden.

2.1.1.2 Dataserver

Unter einem Dataserver wird die Art von Servern verstanden, die die Kommunikation mit der Feldebene ermöglichen und so z.B. den Zugriff auf Daten aus Steuerungen erlauben. In einem RSView SE Projekt können verschiedenste Arten von Dataserver eingebunden werden. Hierbei ist zu beachten, dass es sich um Server handelt, die eine Schnittstelle nach dem OPC Standard bereitstellen. Auch hier ist der Server wieder als logische Einheit zu verstehen, da ein Dataserver aus bis zu zwei physikalischen Computern bestehen kann und es auch möglich ist (aber wieder nicht empfohlen) mehrere Dataserver auf einem Computer zu installieren.

2.1.1.3 RSView SE Client

Der RSView SE Client dient zum Anzeigen der Visualisierung. Um Daten zu beziehen, kommuniziert der Client zur Laufzeit mit HMI und Dataserver. Zum Ermitteln der verfügbaren Server und deren Dienste kommuniziert der Client außerdem noch mit einem Dienst der „FactoryTalk“ genannt wird. Da es sich bei dem Betriebssystem des Clients um ein Windowsbetriebssystem handelt, muss sich auch dieser Computer an der Windowsdomäne anmelden und authentifizieren. Dazu ist auch eine Kommunikation mit dem Domaincontroller erforderlich.

Nachdem der Speicherort eines Displays ermittelt wurde, wird das Display vom Server herunter geladen und lokal auf dem Client ausgeführt. Auch ein eventuell im Display enthaltener VBA Code wird lokal auf dem Client ausgeführt. Dieses Merkmal der Codeausführung ist ein wichtiger Aspekt beim Erstellen von VBA Programmen für RSView SE Displays. Wird z.B. aus dem VBA Programm eine Datenbankverbindung aufgebaut, so geschieht dies von jedem Client aus. Dies muss bei der Programmierung beachtet werden und die Clients müssen bei Bedarf einzeln mit zusätzlich erforderlichen Komponenten ausgestattet werden.

2.1.1.4 RSView Administration Console

Die Konsole dient zur Verwaltung eines angelegten Projektes zur Laufzeit. Es können die Strukturen der Applikation eingesehen werden und einige Serverinformationen abgerufen werden.

2.1.1.5 RSView Studio

Das RSView Studio ist die Entwicklungsumgebung für alle Arten von RSView SE Projekten. Im Studio können die HMI- und Dataserver konfiguriert werden. Außerdem ist die gesamte Struktur der Applikation erkennbar. Neben den HMI- und Dataservern können hier auch so genannte „Areas" erstellt werden, die der logischen Einteilung des Projektes dienen. Die Verwendung von Areas und der Einsatz von mehreren HMI Servern ist ein Merkmal, das nur in „Distributed" Applikationen verwendet werden kann.

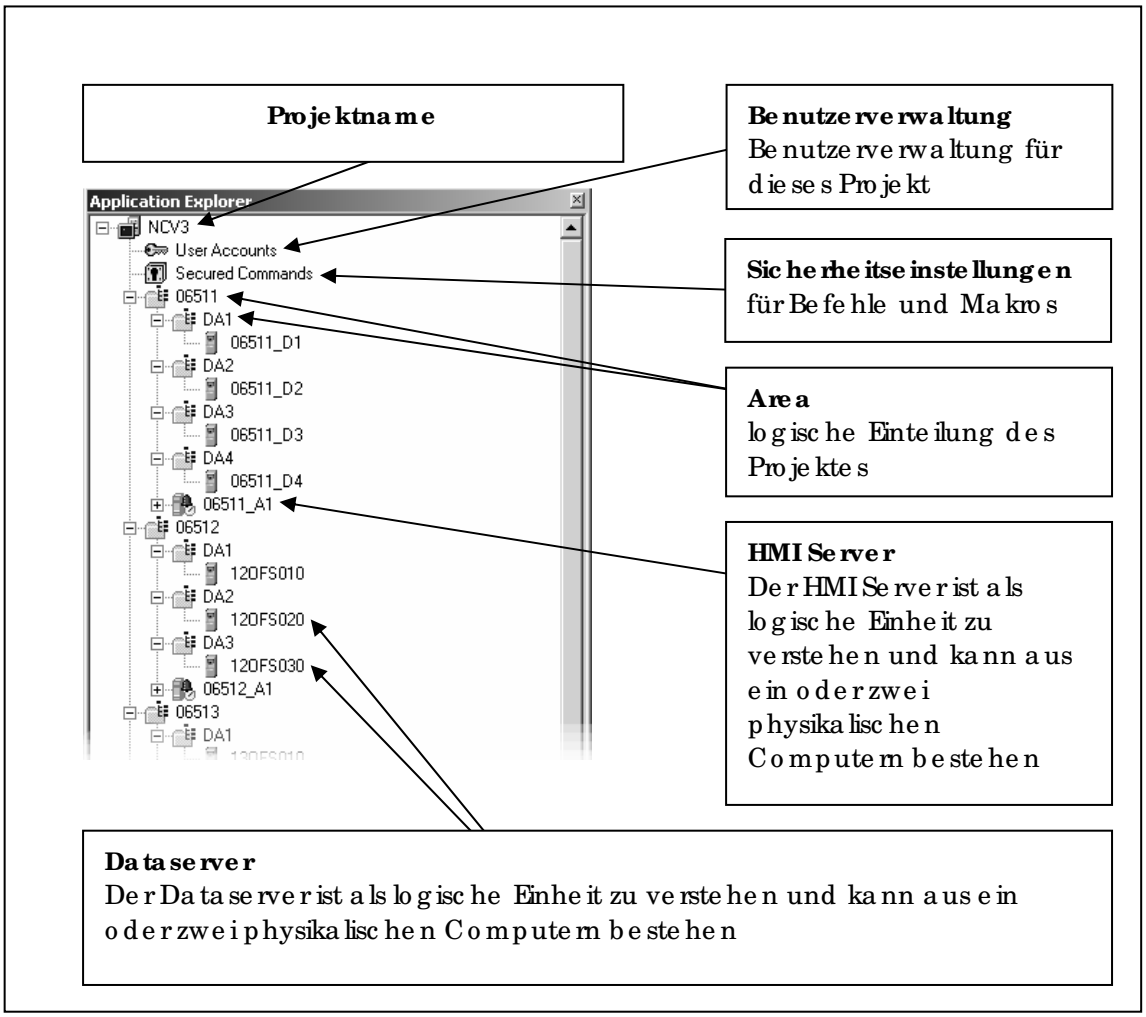


Abb. 2: RSView SE Distributed Applikationstruktur im RSView Studio

Das RSView Studio dient weiterhin auch als Editor für die einzelnen Bilder und weiterer Funktionen von RSView SE. Die Funktionen können nach dem Öffnen eines HMI Servers im Studio bearbeitet werden.

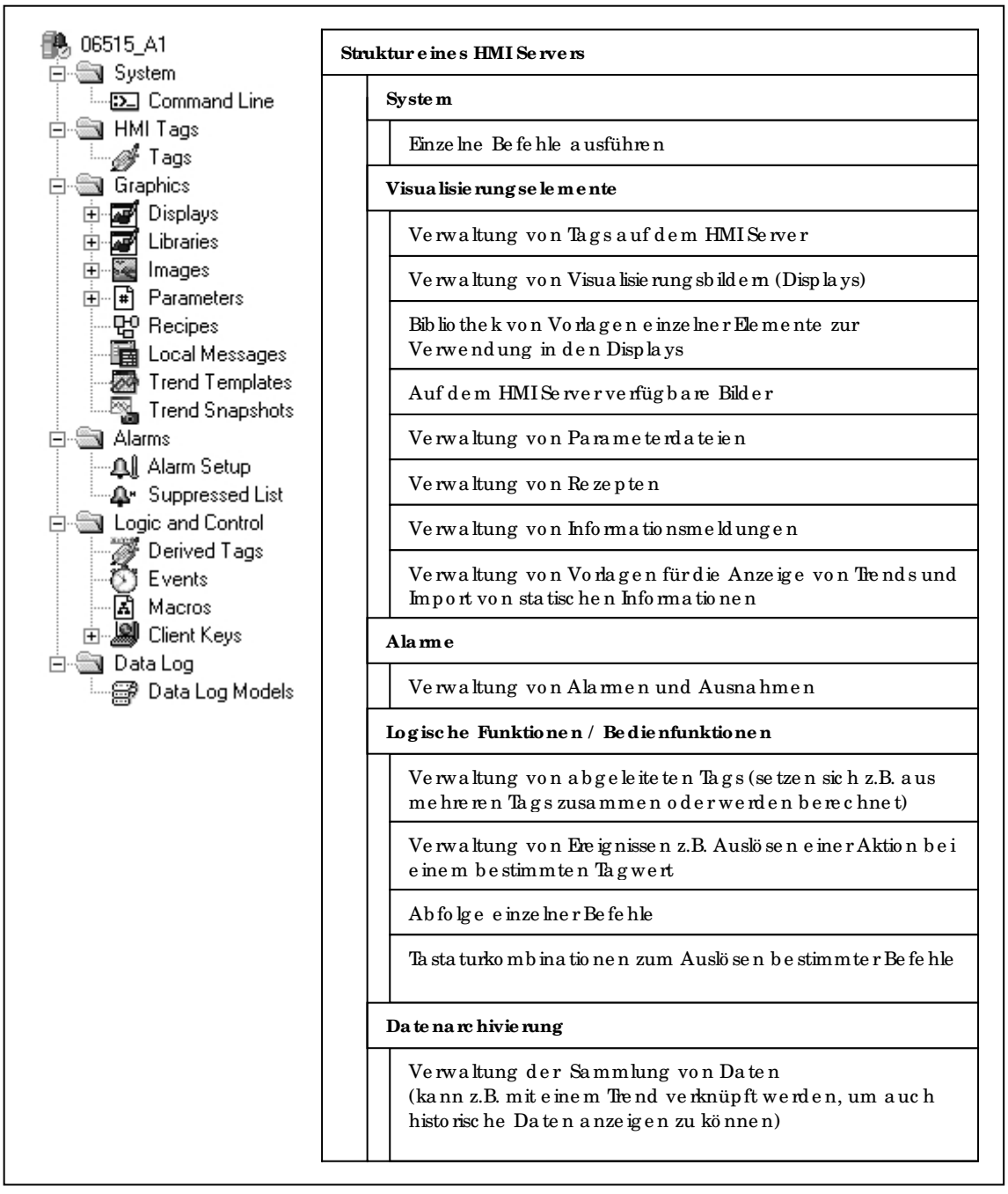


Abb. 3: Struktur eines HMI Servers im RSView Studio

Trotz der vielfältigen Funktionen werden speziell bei der Applikation für die Sprinterproduktion nur die Visualisierungselemente (Dispalys) verwendet. Als Bibliothek dient hierbei nicht die Standardbibliothek, sondern eine Sammlung

an Vorlagen, die speziell für diese Applikation entwickelt wurden. Einige Funktionen wurden im VBA Projekt der einzelnen Displays umgesetzt und werden so unabhängig vom Server ausgeführt.

Neben der Erstellung eines Projektes und einer Verwaltung der Funktionen eines HMI Servers bietet das RSView Studio auch Möglichkeiten, Berechtigungen für bestimmte Benutzer festzulegen und die Einstellungen für das FactoryTalk Eventlog festzulegen.

2.1.2 FactoryTalk

Die Verteilung des Visualisierungssystems erfordert einen Dienst, in dem alle Komponenten verzeichnet sind. Diese Funktion übernimmt das FactoryTalk Verzeichnis. Hierbei handelt es sich um ein System, das allen FactoryTalk basierenden Anwendungen (also nicht nur RSView SE) als zentrales Verzeichnis dienen. Das Gesamtkonzept wird „FactoryTalk Automation Platform“ genannt und beinhaltet folgende Komponenten:

2.1.2.1 FactoryTalk Directory

Da in einem verteilten System mehrere Datenquellen verschiedenster Art existieren können, mussten bei einem klassischen System die einzelnen Datenquellen bzw. deren Datenbanken auf jeden Teilnehmer kopiert werden. Ein anderes Vorgehen wäre eine zentrale Datenbank, die jedoch mit jedem Teilnehmer repliziert, also abgeglichen werden muss. Im Gegensatz dazu werden bei FactoryTalk Informationen über verschiedenste Elemente gesammelt und bereitgestellt. Die Daten an sich verbleiben jedoch in ihrer ursprünglichen Datenquelle, es wird lediglich die Information weitergegeben, an welcher Stelle die gewünschten Daten zu finden sind. Wenn z.B. Daten in einer Steuerung angelegt werden und über einen OPC Server zur Verfügung gestellt werden, der auch auf FactoryTalk zugreifen kann, und außerdem noch einige Bilder auf einem RSView SE HMI Server erstellt wurden, so sind Informationen über diese Elemente in jeder FactoryTalk basierenden Anwendung verfügbar. Hierbei verbleiben die eigentlichen Daten in ihrer ursprünglichen Quelle. Bei der FactoryTalk basierenden Anwendung muss

lediglich der Computername des FactoryTalk Verzeichnisses eingestellt werden, um Daten bereitstellen oder lesen zu können.

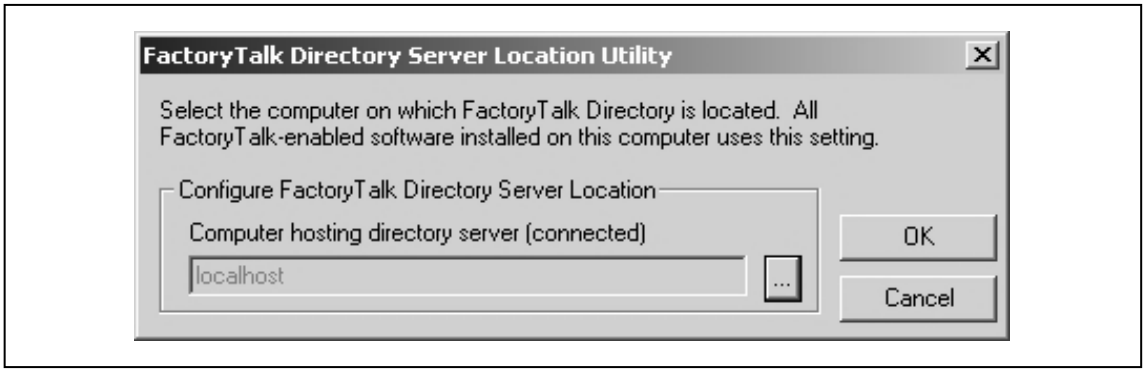


Abb. 4: Konfiguration des FactoryTalk Verzeichnisses

Im Fall einer verteilten Visualisierungsstruktur sind nach dem Angeben des FactoryTalk auf einem Client alle Visualisierungsprojekte verfügbar und es kann das Projekt, das angezeigt werden soll, ausgewählt werden. Wird nun in dem Client ein bestimmtes Display angefordert, so werden folgende Schritte durchlaufen:

1. Beim Aufruf des Displays wird von dem angegebenen FactoryTalk Server die Information abgerufen, auf welchem HMI Server das gewünschte Display gespeichert ist.
2. Nachdem der HMI Server bekannt ist, wird das Display direkt von dem HMI Server abgerufen.
3. Falls das Bild Datenpunkte enthält, die von einem Dataserver geliefert werden, wird wieder vom FactoryTalk die Information abgerufen auf welchem Dataserver die Daten bereitgestellt werden
4. Nachdem der Dataserver ermittelt wurde, werden die Daten direkt vom Client aus bei dem entsprechenden Dataserver angefordert.

2.1.2.2 FactoryTalk Live Data

„FactoryTalk Live Data“ verwaltet die einzelnen Verbindungen von Clients zu Dataserver. Sollte die Verbindung zu einem Dataserver verloren gehen, werden die Clients automatisch benachrichtigt. Außerdem können durch „FactoryTalk Live Data“ mehrere Verbindungen einer Datenquelle zu einer Verbindung für

den Client zusammengeführt werden. Dies ermöglicht im Falle eines Systemausfalls eine Umschaltung auf einen sekundären Server.

2.1.2.3 FactoryTalk Diagnostics

„FactoryTalk Diagnostics“ sammelt Informationen aus dem gesamten System der FactoryTalk basierenden Anwendungen. Hierbei werden Informationen über Ereignisse wie Fehler, Warnungen Änderungen am System oder Informationen in das Windows Eventlog des FactoryTalk Servers geschrieben. Weitere Informationen über einzelne Ereignisse werden in dem Eventlog des jeweiligen Systems gespeichert. Diese Informationen können zur späteren Analyse mit dem „Diagnostics Viewer“ oder einem anderen Eventviewer ausgewertet werden. Aktuelle Meldungen sind auch z.B. in der „Diagnostic List“ von RSView SE sichtbar (falls die Anzeige dieser Leiste zugelassen wird). Diese Meldungen entsprechen den Meldungen, die in das lokale Eventlog geschrieben werden.

2.1.3 RSView SE VBA

Durch die Integration von VBA in RSView SE können innerhalb der Displays eigene VBA Programme erstellt werden. Zusätzlich zu dem integrierten Microsoft Visual Basic 6.3 ist ein RSView SE eigenes „Display Client Object Model“ verfügbar, das mit seinen Funktionen speziell auf die Programmierung für ein RSView SE Display ausgerichtet ist. In diesem Objekt Model sind Klassen speziell mit dem Umgang für die grafischen Elemente des Displays, Tags und dem Display selber hinterlegt. Die Einarbeitung in RSView SE VBA wird durch die - aus anderen Anwendungen bekannte - Entwicklungsumgebung und der zu Visual Basic ähnlichen Syntax deutlich erleichtert.

Für jedes Display ist ein eigenes VBA Projekt vorhanden und ein Display kann näherungsweise als VB Form verstanden werden. So hat ein Button auf einem RSView SE Display ähnliche Funktionalität wie ein Button auf einem VB Form. Für solche Displayelemente stehen standardmäßig einige Ereignisse zu Verfügung, die zum Start der Codeausführung genutzt werden können (im Falle des Buttons wären das unter anderem die Ereignisse „Press“, „Repeat“ oder „Release“). Um

auch eine automatische Codeausführung zu ermöglichen, beinhaltet das Display selber auch Ereignisse die zur Codeausführung genutzt werden können:

- Display_AnimationStart()
- Display_BeforeAnimationStop()
- Display_Load()
- Display_Unload()
- Display_Activate()

Dadurch ist es möglich, unabhängig von Benutzereingaben, VBA Programme anhand des Displays zu starten. Das Ereignis „AnimationStart“ wird zeitlich gesehen kurz nach dem Ereignis „Load“ ausgeführt und eignet sich für ein Programm, das das Display animiert. Das Ereignis „BeforeAnimationStop“ kann zum Schließen von offenen Objekten genutzt werden.

Über die Verwendung der displayeigenen Ereignisse, Buttons und anderen verfügbaren ActiveX Steuerelementen (Microsoft Forms 2.0) kann auch jedes selbst erstellte Objekt wie einfache Striche oder Rechtecke für die Steuerung aus VBA heraus oder sogar zum Auslösen eigener Ereignisse genutzt werden. Insgesamt stehen drei Stufen der Freigabe von Display Objekten in VBA zur Verfügung.

1. **„Not Expose“**
   Objekte mit dieser Einstellung sind nicht direkt durch ein VBA Programm erreichbar
2. **„Type Info Extension“**
   Eigenschaften von Objekten mit dieser Einstellung können aus dem VBA Programm heraus gelesen und geschrieben werden.
3. **„VBA Control“**
   Auf Eigenschaften von Objekten mit dieser Einstellung kann voll zugegriffen werden (wie bei „Type Info Extension“). Außerdem können Eventhandler für diese Objekte erstellt werden. Es sind also Ereignisse für dieses Objekt vorhanden, die genutzt werden können.

Dadurch kann z.B. beim Klicken auf ein bestimmtes Objekt eine Codeausführung gestartet werden.

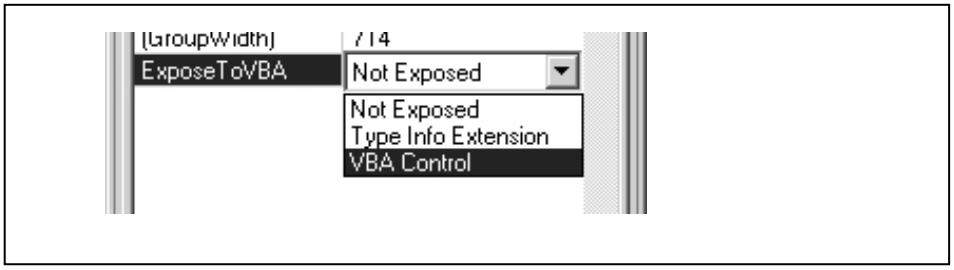


Abb. 5: VBA Einstellung für einzelne Objekte

Bei der Erstellung eines VBA Programms in einem RSView SE Display sind jedoch einige Punkte zu beachten, die vor allem die Stabilität des Programms betreffen.

Ein VBA Programm eines Displays wird in einem einigen Thread abgearbeitet. Dies hat zur Folge, dass bei einer Unterbrechung der Codeausführung solange gestoppt wird bis die Unterbrechung behoben ist. Eine solche Unterbrechung kann neben Fehlern in der Programmierung auch z.B. durch so genannte „Messageboxen“ hervorgerufen werden. Da bei manchen Applikationen das Umschalten zu anderen Fenstern unterdrückt wird, kann eine solche Nachricht zum Teil nicht quittiert werden. Die Unterbrechung bleibt bestehen. Es ist darauf zu achten, dass solche Eigenarten der Programmierung vermieden werden.

Außerdem wird ein vorhandenes Errorhandling vorausgesetzt, damit das Display auch bei einem Fehler in einer Codeausführung bedienbar bleibt. In der Routine, die zur Behandlung der Fehler zuständig ist, sollten alle Fehler abgefangen werden, die Auftreten können. Einzelne Fehler, die zum Teil erwartet werden, können in derselben Routine gesondert behandelt werden. Eine Ermittlung bestimmter Fehler und deren gesonderte Behandlung ist z.B. anhand der Fehlernummer möglich. Insgesamt existieren drei Möglichkeiten um Fehler abzufangen:

- **Inline:** `On Error Resume Next`

  Im Falle eines Fehlers wird der Programmablauf bei dem nächsten Befehl fortgesetzt.
- **Einfache Fehlerbehandlung:** `On Error Goto ErrHandler`

  Beim Auftreten eines Fehler springt das Programm zur Sprungmarke `ErrHandler`. Dort wird eine Fehlerbeschreibung ausgegeben und mit `Resume Next` der Programmablauf fortgesetzt.
- **Bevorzugt:** `On Error Goto ErrHandler`

  Beim Auftreten eines Fehler springt das Programm zur Sprungmarke `ErrHandler`. An dieser Stelle kann nun zwischen verschiedenen Fehlern unterschieden werden. Es sollten ausführliche Informationen über den Fehler gespeichert werden (Fehlernummer, Beschreibung, Zeitstempel, Ursprung). Außerdem sollte zwischen einem Programmablauf im Testbetrieb und einem Programmablauf im normalen Betrieb unterschieden werden. Im Testbetrieb könnte eine „Messagebox“ aufgerufen werden, um einen Programmabbruch zu ermöglichen. Dieses Vorgehen sollte aber aus den oben genannten Gründen im normalen Betrieb vermieden werden.

2.2 OPC

OPC ist eine Standardschnittstelle zum Zugriff auf Daten in der Feldebene. Die OPC Foundation ist 1996 aus der OPC Task Force – ein Zusammenschluss mehrere Unternehmen wie z.B. Fisher-Rosemount, Rockwell Software und Intellution hervorgegangen, die die ersten OPC Spezifikationen veröffentlichten *[2.2]*.

Ein beliebter Vergleich für den Zweck von OPC ist die Verwaltung von Druckern unter Microsoft DOS und Windows. Unter DOS Systemen musste der Hersteller einer Software auch Treiber für Drucker bereitstellen. Unter Windows wurde eine einheitliche Schnittstelle für Drucker eingeführt auf die jede Software zugreifen kann. Ein direkter Zugriff auf den Drucker - und somit die Entwicklung eigener Druckertreiber - ist für die einzelnen Softwarehersteller nicht mehr nötig, die Verwaltung liegt bei dem Betriebssystem.

Ähnlich verhält es sich mit dem Zugriff auf Daten aus der Feldebene. Ohne eine Standardschnittstelle mussten die Hersteller von eigenen Treibern für den Zugriff auf z.B. Steuerungen entwickeln. Ein problemloser Austausch z.B. eines Visualisierungssystems war nur möglich, wenn es für das neue System entsprechende Treiber für die Steuerungen gab. Durch die einheitliche Schnittstelle wird der Zugriff auf Daten der Feldebene deutlich erleichtert und ermöglicht auch einen problemlosen Datenaustausch zwischen Systemen verschiedener Hersteller.

Durch dieses Vorgehen kann ein Hersteller über einen Standard OPC Client auf die Daten der Feldebene zugreifen (werden bereitgestellt von einem OPC Server).

OPC basiert auf den Microsoft Protokollen COM (Component Objekt Model) und DCOM (Distributed Component Object Model), die es Programmen und Programmkomponenten erlaubt, miteinander zu kommunizieren.

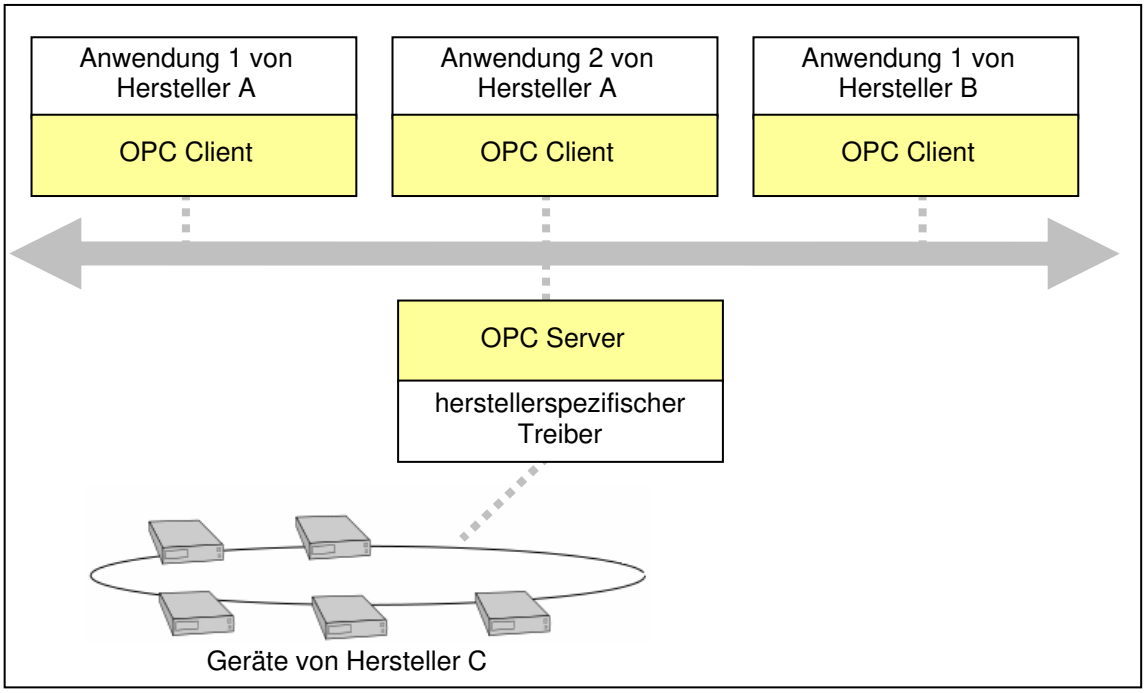


Abb. 6: Schema Datenzugriff mittels OPC

Das zur Verwaltung der Startvorgänge von OPC Servern benötigte Programm ist der „OPC Server Enumerator". Dieses Programm stellt Informationen über einen oder mehrere OPC Server, die auf einem Computer installiert sind, zur Verfügung. Entwickelt wurde dieses Programm von der OPC Foundation, wird aber von dem Hersteller des OPC Servers mit ausgeliefert *[2.3]*.

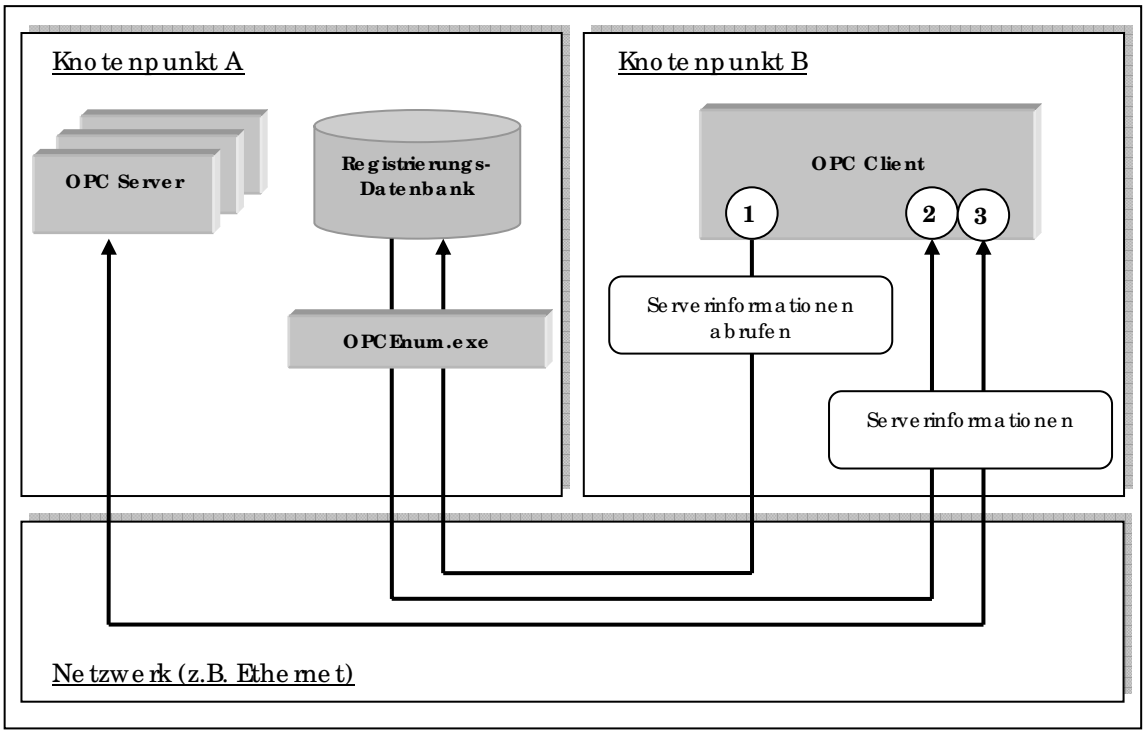


Abb. 7: OPC Server Zugriff

Die oben dargestellte Grafik zeigt den Start der Kommunikation von einem Client mit einem OPC Server:

1. Aufruf zum Durchsuchen („browsen“) des OPC Servers.
2. Eine OPC Serverliste der installierten OPC Server wird aus der Registrierungsdatenbank des Servers gelesen und an den Client zurückgegeben.
3. Nach der Ermittlung der verfügbaren Server kann der Client direkt den gewünschten Server ansprechen.

2.3 Ethernet

Die fortschreitenden Anforderungen an Computernetzwerke in den Bereichen Bürokommunikation, technische und wissenschaftliche Anwendungen, Konstruktion, Fertigung hat seit dem Jahr 1972 die Entwicklung der Ethernets vorangetrieben. Seit dieser Zeit wurde ein Netzwerk entwickelt, das auf dem CSMA/CD Zugriffsverfahren basiert. Insgesamt setzt sich die Datenübertragung nach diesem Verfahren aus drei Schritten zusammen:

- **C**arrier **S**ense
  Jeder Netzteilnehmer prüft, ob das Übertragungsmedium frei ist
- **M**ultiple **Ac**cess
  Bei freiem Übertragungsmedium kann jeder Netzteilnehmer beginnen, Daten zu senden
- **C**ollision **D**etection
  Beginnen mehrere Netzteilnehmer gleichzeitig Daten zu senden, kommt es zu einer Überlagerung der einzelnen Signale. Dies wird als Datenkollision bezeichnet. Da bei einer Überlagerung von Signalen der Signalpegel ansteigt, kann daran eine Kollision erkannt werden. Es werden von dem Netzwerkteilnehmer entsprechende Maßnahmen eingeleitet

Die Arbeitsgruppe 802 der „Institution of Electronical and Electronic Engineers“ (IEEE) hat aus diesem Verfahren einen Standard erarbeitet. Die Vorlage der Empfehlung IEEE 803.3 bei der International Standardization Organisation / International Electrotechnical Commission (ISO/IEC) führte zu der Norm ISO/IEC 8802-3 *[2.4]*.

Ein weiterer bedeutender Schritt bei der Entwicklung des Ethernets war die Entwicklung eines Referenzmodells der verschiedenen Netzwerkschichten. Dieses Referenzmodell ist als ISO/OSI-Referenzmodell (ISO / Open System Interconnection) bekannt und verfolgt folgende Ziele:

- einen Standard für den Informationsaustausch zwischen offenen Systemen zu definieren
- eine gemeinsame Basis für die Entwicklung von weiteren Standards für offene Systeme zur Verfügung zu stellen
- internationale Expertenteams mit einem funktionellen Gerippe zur unabhängigen Entwicklung für jede Schicht des Modells zu versorgen
- schon bestehende oder in der Entwicklung befindliche Protokolle zur Kommunikation verschiedener Systeme untereinander in diesem Modell zu berücksichtigen
- genügend Raum und Flexibilität für zukünftige Erweiterungen zu zulassen

In diesem Referenzmodell werden verschiedene Schichten von der Bitübertragungsschicht zur Anwendungsschicht beschrieben. Die Veröffentlichung dieses Modells erfolgte im Oktober 1984 im internationalen Standard ISO 7498 *[2.5]*.

OSI Schicht		**Beschreibung**	
7	Anwendung	Aus einem Anwenderprogramm auf Kommunikationsdienste zugreifen	Anwendungs-orientiert
6	Darstellung	Definition der Syntaxdarstellung für den Datenverkehr	
5	Sitzung	Auf- und Abbau von Verbindungen durch Synchronisation und Organisation des Dialogs	
4	Transport	Festlegung der Endsystemverbindung mit der erforderlichen Transportqualität	Transportorientiert
3	Vermittlung	Transparenter Datenaustausch zwischen zwei Transporteinheiten	
2	Sicherung	Zugang zum physikalischen Medium, sowie Erkennen und Beheben von Übertragungsfehlern	
1	Bitübertragung	Übertragung von Bitströmen auf physikalisch vorhandenen Medien	

Abb. 8: OSI-Referenzmodell

Für die weit verbreitete Internet-Protokoll Familie wird ein Schichtmodell verwendet, das im Gegensatz zum OSI-Referenzmodell auf vier Schichten reduziert wurde. Der Zugriff auf das Medium wird in diesem Fall nicht weiter beschrieben.

TCP/IP Schicht		Entspricht OSI Schicht	Beispiel
4	Anwendung	7 Anwendungsschicht 6 Darstellungsschicht 5 Sitzungsschicht	HTTP
3	Transport	4 Transportschicht	TCP, UDP
2	Vermittlung	3 Vermittlungsschicht	IP
1	Netzzugang	2 Sicherungsschicht 1 Bitübertragungsschicht	Ethernet

Abb. 9: TCP/IP Referenzmodell

Im Laufe der Entwicklung wurde das Grundprinzip beibehalten. Die Datenrate und die Übertragungsmedien änderten sich jedoch. 1992 wurde mit der Entwicklung des 100Mbit/s Ethernet begonnen. Für dieses Netzwerk existierten seit 1994 auch Lösungen zum Einsatz von Twisted-Pair Kabeln. Zunächst konkurrierten zwei Lösungsansätze. Beim Fast-Ethernet wurden die Übertragungsparameter an die neue Geschwindigkeit angepasst. Der große Nachteil dieses Verfahrens war, dass die mögliche Ausdehnung stark reduziert wurde. Die zweite Variante, die ein anderes Zugriffsverfahren benutze und von der IEEE durch die Projektgruppe 802.12 verabschiedet wurde, konnte sich nicht durchsetzen. Das Fast-Ethernet und die schnellere Variante Gigabit-Ethernet sind heute Standard und noch schnellere Varianten sind in Entwicklung.

Der große Nachteil des Fast-Ethernets wurde durch die schnelle Weiterentwicklung von Brücken und Switchen unbedeutend.

2.4 VBA - Visual Basic for Applications

Visual Basic for Applications ist eine von Microsoft entwickelte Programmiersprache zur Erweiterung und Steuerung von Programmen. Ursprünglich wurde diese Erweiterung in den Microsoft Office Programmen eingesetzt und ermöglichte dem Benutzer mit einer im gesamten Microsoft Office Programm einheitlichen Sprache eigene Programmabläufe zu erstellen. Die Syntax ähnelt der weit verbreiteten Programmiersprache Visual Basic und erleichtert so den Einstieg in die VBA Programmierung *[2.6]*.
Neben den Funktionen des VBA Kerns, wie die Verfügbarkeit verschiedener Datentypen, Kontrollstrukturen (verschiedene Schleifenfunktionen) und Zugriff auf das Dateisystem, steht auch ein, der Anwendung in der VBA ausgeführt wird angepassten, Objekt Modell zur Verfügung. In RSView SE wird dieses Objektmodell „Display Client Object Model" genannte und bietet Zugriff auf spezielle Klassen, die auf RSView SE ausgerichtet sind.

Die Möglichkeiten und der Komfort der Entwicklungsumgebung von VBA sind im Gegensatz zu Visual Basic oder Visual Basic .NET eingeschränkt. Trotzt dieser Einschränkung entstehen durch Hinzufügen weiterer Objektklassen umfangreiche Möglichkeiten. So können Referenzen eingebunden werden, die den Umgang mit XML Dateien ermöglichen, den Zugriff auf die Windowsverwaltung und Verbindungen zu verschiedensten Datenbanken ermöglichen. Dies sind nur einige wenige Beispiele, sie wurden jedoch im Rahmen dieser Arbeit genutzt. Außerdem besteht die Möglichkeit, Funktionen aus anderen Dateien zu benutzen (siehe Kapitel „API - Application Programming Interface").

Alle in dieser Arbeit aufgeführten Codebeispiele sind in RSView SE VBA oder VBA für Excel entstanden.

2.5 API - Application Programming Interface

Unter API wird im Allgemeinen eine Schnittstelle verstanden, die ein Softwaresystem für den Zugriff zur Verfügung stellt.
Dank dieser Technik, die auf Windowssystemen weit verbreitet ist, ist ein Zugriff auf Funktionen des Betriebssystems und anderer Anwendungen möglich.

Hierzu ein Anwendungsbeispiel:
In einem VBA Programm (in welcher Umgebung das Programm läuft, ist für dieses Beispiel nicht von Bedeutung) soll ein Timer verwendet werden. Im Gegensatz zu anderen Umgebungen steht in VBA aber keine Timer Komponente zur Verfügung. In der Windows Systemdatei „user32.dll" wird jedoch ein Timer zur Verfügung gestellt, der auch in VBA verwendet werden kann. Bei dieser Art der Verwendung (Zugriff auf eine Funktion in einer Dynamik Link Library) handelt es sich um eine „funktionsorientierte Programmierschnittstelle" *[2.7]*.

```
Public Declare Function SetTimer Lib "user32" _
  (ByVal HWnd As Long, ByVal nIDEvent As Long, _
  ByVal uElapse As Long, ByVal lpTimerFunc As Long) As Long
```

VBA Code 1: Deklaration der SetTimer Funktion in VBA

Durch Verwendung dieser Funktion wird nach dem in Millisekunden angegebenen Intervall `uElapse` ein Ereignis vom Betriebssystem ausgelöst, das die angegebene Funktion `lpTimerFunc` (Adresse der Funktion) startet. Der Parameter `HWnd` gibt das zu dem Timer gehörenden Fenster an. Sollte ein solcher Bezug nicht existieren oder nicht gewünscht sein, kann NULL übergeben werden. In solch einem Fall wird der Parameter `nIDEvent` ignoriert, andernfalls müsste hierein "Timeridentifier" angegeben werden.
Bei diesem Beispiel ist darauf zu achten, dass der Timer nach Verwendung wieder beendet wird, weil das Ereignis vom Betriebsystem ansonsten trotzdem ausgelöst wird und zu einem Fehler führt, da die angegebene Funktion nicht mehr existiert *[2.8]*.

Auch hierzu kann eine Funktion der „user32.dll“ verwendet werden:

```
Public Declare Function KillTimer Lib "user32" _
  (ByVal HWnd As Long, ByVal nIDEvent As Long) As Long
```

VBA Code 2: Deklaration der KillTimer Funktion zum Beenden des Timers

Auch bei dieser Funktion kann für den Parameter `HWnd` der Wert NULL angegeben werden. Mit dem Parameter `nIDEvent` wird der Timer angegeben, der beendet werden soll.

Durch diese beiden Deklarationen kann nun ein Timer aus einem anderen Programm in dem VBA Programm verwendet werden:

```
Public Function StartTimer()
  On Error GoTo Errorhandler
  Dim lngInterval As Long
  lngInterval = 2000
  lngTimerID = SetTimer(0&, 0&, lngInterval&, _
                        AddressOf TimerTriggered)

  Exit Function
Errorhandler:
    MsgBox "(" + CStr(Err.Number) + ") " + Err.Description
End Function
```

VBA Code 3: Funktion zum Starten des Timers

```
Public Function TimerTriggered(ByVal HWnd As Long, _
                               ByVal uMsg As Long, _
                               ByVal nIDEvent As Long, _
                               ByVal dwTimer As Long)
  On Error GoTo Errorhandler
    MsgBox "Timer ausgelöst!"
  Exit Function
Errorhandler:
    MsgBox "(" + CStr(Err.Number) + ") " + Err.Description
End Function
```

VBA Code 4: Funktionsaufruf sobald das Interval abgelaufen ist

```
Public Function EndTimer()

    On Error Resume Next
    KillTimer 0&, lngTimerID

End Function
```

VBA Code 5: Beenden des Timers

Durch diese Methode ist es möglich, eine Vielzahl von Funktionen, die Windows und andere Programme bereitstellen, auch in VBA zu benutzen.

2.6 WMI – Windows Management Instrumentation

Das Windows-Verwaltungsinstrumentation WMI (Windows Management Instrumentation) ist die Microsoft-Implementierung von WBEM (Web-Based Enterprise Management, Webbasierendes Unternehmensmanagement), einer Industrieinitiative zur Standardisierung des Zugriffs und der Freigabe von Verwaltungsinformationen in einem Unternehmensnetzwerk. WMI ist WBEM-kompatibel und bietet integrierte Unterstützung für CIM (Common Information Model, Gemeinsames Informationsmodell), einem Datenmodell zur Beschreibung der Objekte in einer Verwaltungsumgebung an *[2.9]*.

Die WMI-Komponente umfasst ein CIM-kompatibles Objektrepository, d. h. eine Datenbank mit Objektdefinitionen, und den CIM-Objekt-Manager, der die Zusammenstellung und Bearbeitung der Objekte im Repository („Lager") verwaltet und Daten von den WMI-Anbietern sammelt. WMI-Anbieter fungieren als Vermittler zwischen WMI und Betriebssystemkomponenten, Anwendungen und anderen Systemen. Beispielsweise bezieht der Registrierungsanbieter Daten aus der Registrierung, während der SNMP-Anbieter Daten und Ereignisse von SNMP-Geräten liefert. Anbieter stellen Informationen zu ihren Komponenten bereit und möglicherweise auch Methoden zum Manipulieren der Komponenten, Eigenschaften, die festgelegt werden können, oder Ereignisse, die sie auf Änderungen an den Komponenten hinweisen.

WMI kann von Computerverwaltungstools wie z. B. Microsoft Systems Management Server zur Unterstützung bei der Verwaltung von Computern verwendet werden. WMI wird auch von anderen Microsoft-Technologien und -Tools, z. B. Microsoft Health Monitor und Microsoft Operations Manager, sowie von anderen Herstellern von Computerverwaltungssystemen verwendet. Durch WMI Programmier- oder Skriptingsystemen (z. B. Windows Script Host) können Skripte verwendet werden, um Konfigurationsdetails zu den meisten Aspekten des Computersystems einschließlich Serveranwendungen abzurufen oder um Änderungen an Ihren Systemen vorzunehmen.

Eine Reihe von Verwaltungsprogrammen unterstützen WMI, darunter Systemeigenschaften, Systeminformationen und die Komponente Abhängigkeiten in den Diensten. Diese Komponenten werden im Folgenden kurz erläutert:

Über das Programm „Systemeigenschaften“ können Systemeigenschaften auf einem lokalen Computer oder Remotecomputer betrachtet und geändert werden. Mit diesem Programm kann Remotecomputer neu gestartet werden, um geänderte Einstellungen zu übernehmen oder neue Hardware zu ermitteln. Außerdem kann der Computernamen und Informationen zur Domäne anderer Computer im Netzwerk anzeigen sowie die Einstellungen der Auslagerungsdatei des virtuellen Speichers auf einem Computer, auf dem speicherintensive Programme ausgeführt werden, geändert werden. Mithilfe von Systeminformationen werden Informationen zur Systemkonfiguration gesammelt und angezeigt. Dies ist besonders nützlich, wenn Fehler behoben werden sollen *[2.10]*.

Zur manuellen Analyse der zur Verfügung gestellten Daten bietet Microsoft eine Sammlung von Programmen an (Microsoft WMI Administrative Tools), die frei über die Webseite von Microsoft bezogen werden können.

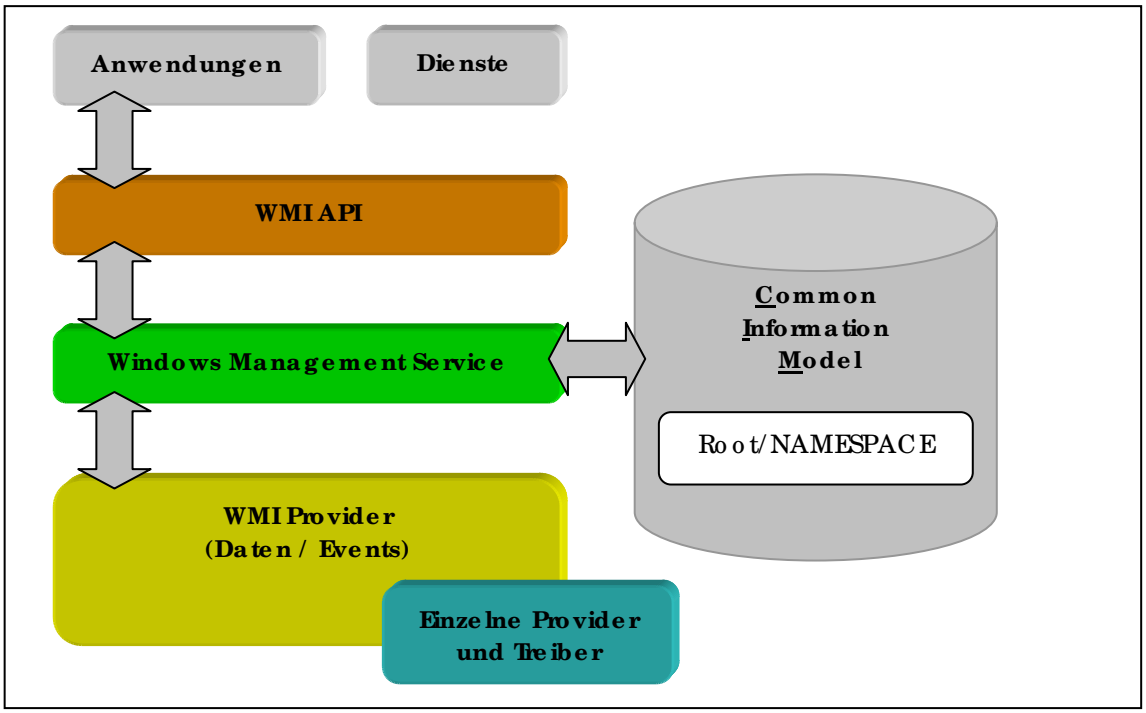


Abb. 10: WMI Architektur

3 Aufgabenstellung

In einer weit verteilten Visualisierungsapplikation wie sie im Zusammenhang mit der Sprinterproduktion von DaimlerChrysler im Werk Düsseldorf eingesetzt wird, ist es notwendig, die Anlage zu überwachen. Die Überwachung und Steuerung der Anlage übernimmt die Visualisierungsapplikation selbst. Eine Überwachung der Visualisierung ist ein weiterer Schritt, um einen möglichst ausfallfreien Betrieb zu gewährleisten bzw. Fehler schnell beheben zu können. Für diesen Zweck werden verschiedene Programme zur Verfügung gestellt, um die einzelnen Komponenten diagnostizieren zu können. So kann über die „RSView SE Administration Console“ die Visualisierungsapplikation an sich überwacht werden und z.B. der Status der HMI Server abgerufen werden. Um den Status der einzelnen Dataserver zu ermitteln muss, ein entsprechendes Programm des OPC Server Herstellers benutzt werden. Das Netzwerk, welches in diesem Fall aus Managed Switches der Firma Hirschmann besteht, kann über eine Netzwerkmanagementsoftware der Firma Hirschmann überwacht und gewartet werden.
Um einen schnellen Überblick über den Gesamtstatus zu bekommen, sind diese Einzellösungen jedoch zu aufwändig und erfordern vom Benutzer zusätzliche Kenntnisse über die spezielle Anwendung. Sinnvoll wäre es, in der Visualisierungsapplikation selber Informationen bereitzustellen. Die Vorteile eines solchen Vorgehens wären, dass die wichtigsten Informationen in einer vertrauten Umgebung dargstellt werden können und, aufgrund des verteilten Systems, auf allen HMI Clients eingesetzt werden können.
Ziel dieser Arbeit ist es, die Möglichkeiten der Informationssammlung, Verarbeitung und Darstellung der verschiedenen Statusinformationen zu untersuchen und in RSView SE Displays darzustellen. Ausgangspunkt dieser Arbeit ist eine früher erstellte Visualisierung, die dem Zweck hat Daten aus der Anlage und der Leittechnik gegenüber zu stellen. In Grundzügen standen auch hier schon einige Diagnoseinformationen bereit. Die einzelnen Funktionen müssen jedoch zum großen Teil neu erstellt werden, um die gewünschte Funktionalität zu erreichen.

Zu diesem Zweck muss zunächst untersucht werden überwelche Komponenten sinnvollerweise Informationen angezeigt werden sollen und auf welchem Weg

diese Informationen gesammelt werden können. Aufgrund des Windows-Netzwerkes, in dem die Visualisierung betrieben wird, und den vielfältigen Möglichkeiten der VBA Programmierung innerhalb der RSView SE Displays stehen eine Reihe von Techniken zur Informationssammlung und Auswertung zur Verfügung.

Aus dieser Sammlung verschiedenster Möglichkeiten müssen die geeigneten ausgewählt werden und Funktionen gegeneinander abgewogen werden.

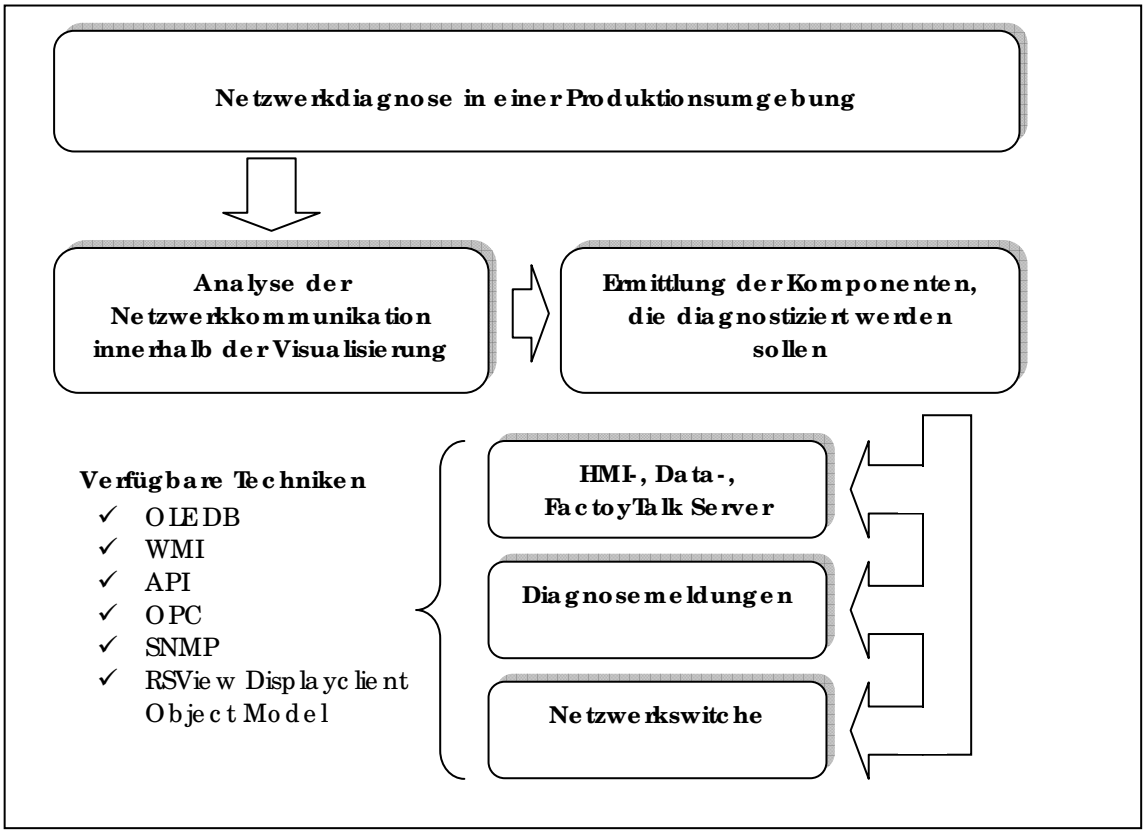


Abb. 11: Struktur der Aufgabe

4 RSView SE Client – Server Kommunikation

Um eine Aussage über die Komponenten treffen zu können, mit denen ein RSView SE Client üblicherweise kommuniziert, wurde der Netzwerkverkehr eines Clients aufgezeichnet und analysiert.
Die so gewonnenen Daten dienen der Auswahl der zu diagnostizierenden Netzwerkkomponenten.

4.1 Vorgehen der Aufzeichnung

Um den Netzwerkverkehr aufzuzeichnen wurde ein Paketsniffer verwendet. Die Aufzeichnung des Netzwerkverkehrs erfolgt an der Netzwerkkarte eines RSView SE Clients.
Da es sich bei dem RSView SE Client um einen Windows PC handelt und die Aufzeichnung in einem extra zu diesem Zweck aufgebauten Netzwerk erfolgt, konnte der Paketsniffer direkt auf dem Client installiert werden. Sollte ein solches Vorgehen nicht möglich sein, da z.B. auf dem Client keine Software installiert werden darf oder es sich bei dem Gerät, zu dem der Netzwerkverkehr aufgezeichnet werden soll, um ein Gerät handelt auf dem keine Installation von Software möglich ist (SPS, Switch, usw.), empfiehlt sich der Einsatz von einem Switch, der Port Mirroring unterstützt oder die Verwendung eines Hubs, der zwischen dem Gerät und dem ursprünglichen Hub oder Switch eingesetzt wird.
Das hier zur Aufzeichnung verwendete Tool ist der Netzwerk Protokoll Analyser „Wireshark“ (ehemals „Ethereal“).

4.2 Aufbau des Testnetzwerkes zur Analyse des Netzwerkverkehrs

In dem verwendeten Netzwerk sind folgende Komponenten enthalten:

Typ	Netzwerkname	IP
Domain Controller	DC_AT_PRI	192.168.0.1
FactoryTalk Server	ROCKWELL_FTD	192.168.0.4
HMI Server	HMI_SERVER	192.168.0.2
HMI Server	HMI_SERVER_SEC	192.168.0.6
Dataserver	ROCKWELL_DATA	192.168.0.3
Dataserver	ROCKWELL_DATA_SEC	192.168.0.7
Datenbank + Soft SPS	SQL_Server	192.168.0.5
RSView SE Client	WinXP1	192.168.0.11
RSView SE Client	1120002WBS010	192.168.0.12
Unmanaged Switch	--	--

Abb. 12: Testnetzwerk Konfiguration

Das Netzwerk besteht auf Grund der aktivierten Redundanz aus einem HMI Server und einem Dataserver, der Daten von einer Soft SPS liest. Als Clients stehen zwei Systeme zur Verfügung.

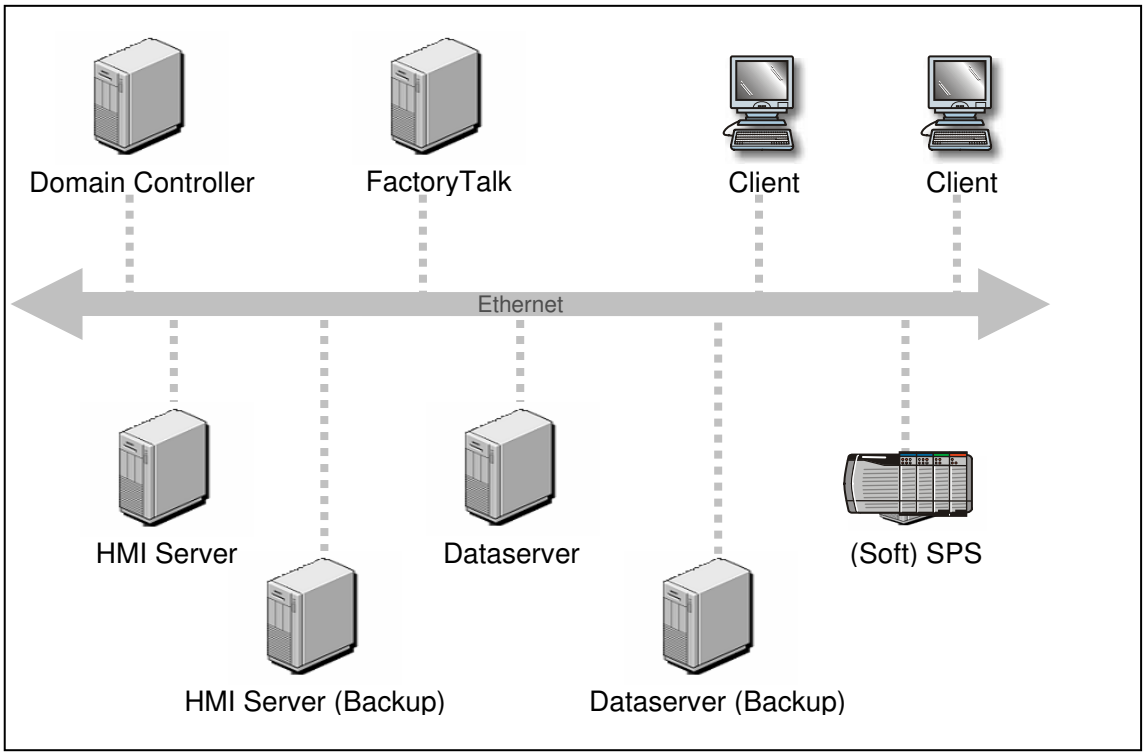


Abb. 13: Testnetzwerk Aufbau

Die installierten Systeme sind Microsoft Virtual PCs. Dadurch ist ein Höchstmass an Flexibilität beim Konfigurieren der einzelnen Systeme gewährleistet, ohne dass ein echtes System beeinflusst wird.
Durch dieses Vorgehen können auch einzelne Systeme auf andere Host Computer übertragen werden, ohne die Funktion des virtuellen Systems zu beeinflussen.

4.3 Analyse des Netzwerkverkehrs

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.12	192.168.0.4	ICMP	Echo (ping) request
2	0.002148	192.168.0.4	192.168.0.12	ICMP	Echo (ping) reply
3	0.003024	192.168.0.12	192.168.0.3	ICMP	Echo (ping) request
4	0.005391	192.168.0.3	192.168.0.12	ICMP	Echo (ping) reply
5	0.012566	192.168.0.12	192.168.0.2	ICMP	Echo (ping) request
6	0.017329	192.168.0.2	192.168.0.12	ICMP	Echo (ping) reply
7	0.109637	192.168.0.2	192.168.0.255	NBNS	Name query
8	0.930888	192.168.0.2	192.168.0.255	NBNS	Name query
9	1.000791	192.168.0.12	192.168.0.4	ICMP	Echo (ping) request
10	1.002740	192.168.0.4	192.168.0.12	ICMP	Echo (ping) reply
11	1.003648	192.168.0.12	192.168.0.3	ICMP	Echo (ping) request
12	1.009805	192.168.0.3	192.168.0.12	ICMP	Echo (ping) reply
13	1.010229	192.168.0.12	192.168.0.2	ICMP	Echo (ping) request
14	1.011240	192.168.0.2	192.168.0.12	ICMP	Echo (ping) reply
15	1.944899	Microsof_6d:bc:89	Broadcast	ARP	Who has 192.168.0.6?
16	2.006254	192.168.0.12	192.168.0.4	ICMP	Echo (ping) request
17	2.007486	192.168.0.4	192.168.0.12	ICMP	Echo (ping) reply
18	2.008118	192.168.0.12	192.168.0.3	ICMP	Echo (ping) request
19	2.011423	192.168.0.3	192.168.0.12	ICMP	Echo (ping) reply
20	2.011802	192.168.0.12	192.168.0.2	ICMP	Echo (ping) request

Abb. 14: Netzwerkverkehr eines RSView Clients im Ruhezustand

Aus dem aufgezeichneten Netzwerkverkehr geht hervor, dass der RSView SE Client (Computer mit der IP 192.168.0.12) der Reihe nach die Server „anpingt“, die in dem angelegten RSView Projekt vorhanden sind, also FactoryTalk (IP 192.168.0.4), Dataserver (IP 192.168.0.3) und den HMI Server (IP 102.168.0.2). Nach jeder Ping-Anfrage empfängt der Client eine Antwort von dem entsprechenden Server. Sollte eine Antwort ausbleiben, ist dies das Zeichen, dass der Server ausgefallen ist. Mögliche Gründe hierfür reichen im Prinzip quer durch das Netzwerkschichtmodells und können z.B. eine physikalische Unterbrechung der Leitung sein, eine ungültige oder keine IP Adresse oder ein nicht reagierender Dienst auf dem Zielcomputer. Eine ausbleibende Antwort würde in dem FactoryTalk Log des Clients verzeichnet werden und eine Redundanzumschaltung auslösen.

Neben den Ping-Anfragen sind im Netzwerkprotokoll weitere Kommunikationen zu finden, die aber nicht zu der Kommunikation aus RSView gehören. Im Paket (7) und (8) werden vom FactoryTalk Server aus Anfragen zur Namensauflösung verschickt. Hierfür wird das „Network Basic Input Output System Name Server Protokoll“ (kurz NBNS) verwendet und eine Anfrage über Broadcast verschickt. Diese Anfrage ist also an alle Netzwerkteilnehmer in dem Netz gerichtet und erreicht deshalb auch den RSView SE Client, obwohl dieser PC nicht betroffen ist und auch keine Antwort gesendet hat. Bei genauerer Betrachtung des Pakets ist zu erkennen, dass die IP Adresse zu dem Computernamen „ROCKWELL_DATA_S“ (sekundärere Dataserver, der aber in dem, auf dem RSView SE Client aufgerufenen Projekt, nicht verwendet wird) ermittelt werden soll.

Das Paket (15) dient ebenfalls der Adressauflösung, allerdings soll dieses Mal eine IP Adresse 192.168.0.6 mittels des „Adress Resolution Protokolls“ (ARP) in eine Ethernetadresse (MAC – Media Access Control) aufgelöst werden. Der Aufruf kam in diesem Fall wieder von dem FactoryTalk Server (MAC Adresse: 00:03:ff:6d:bc:89) und richtete sich als Broadcast wieder an alle erreichbaren Netzwerkteilnehmer. Da auch in diesem Fall der Client nicht von dem Aufruf betroffen ist, wird die Information über MAC Adresse und zugehöriger IP Adresse des Absenders in seinem ARP-Cache aktualisiert bzw. gespeichert aber ansonsten keine weitere Aktion ausgeführt.

Ein Displayaufruf kann an einer vom RSView SE Client ausgehenden HTTP (Hypertext Tranfer Protocol) erkannt werden.

```
No. Time      Source              Destination    Protocol
33 3.043824  192.168.0.12        192.168.0.2    HTTP
                                 GET /rsviewse/cc_hmi1/gfx/display2.gfx HTTP/1.1

36 3.046750  192.168.0.2        192.168.0.12   http
                                 HTTP/1.1 304 Not Modified
```

Abb. 15: HTTP Aufruf eines Displays

Die Anfrage wird vom RSView SE Client (IP 192.168.0.12) aus direkt an den HMI Server (IP 192.168.0.2) geschickt. Als Antwort schickt der Server in diesem Fall einen HTTP Status Code 304, der bedeutet, das sich die angeforderte Datei nicht verändert hat. Jede Anfrage an den HMI Server über HTTP wird mit solch einem Statuscode beantwortet. Derartige Anfragen werden von dem Webserver (in diesem Fall Microsoft Internet Information Service), der auf dem HMI Server installiert ist, bearbeitet. Weit verbreitet beim „Surfen" im Internet sind z.B. die Statuscodes 404 wenn eine Datei nicht gefunden werden kann oder der Statuscode 403 wenn der Zugriff auf eine bestimmte Datei nicht erlaubt ist. Beim Verteilen der RSView Bilder werden dieselben Mechanismen verwendet.

4.4 Schlussfolgerungen aus der Analyse des Netzwerkverkehrs

Bei der Betrachtung des Netzwerkverkehrs ist festzustellen, dass ein RSView SE Client mit allen Servern, die im Projekt angelegt sind, kommuniziert. Auch in dem Fall, das für einige Zeit keine Daten von einem Server benötigt werden, (so werden in dem Beispiel keine Daten vom Dataserver bezogen) werden die Server dennoch durch nahezu kontinuierliches „anpingen" von jedem Client aus überwacht.

Um einen Gesamtstatus aller beteiligten Komponenten zu bekommen, muss also der Status sämtlicher Server, die in einem RSView Projekt angelegt sind, überwacht werden.

5 Server Status

Die wesentlichen Komponenten mit denen ein RSView Client kommuniziert sind die in der Visualisierung angelegten Server. Diese Server sind von den Typen:

- HMI Server
- Dataserver (evtl. von einem Drittanbieter)
- FactoryTalk Server

Da die empfohlene Umgebung der Visualisierung mit RSView eine Windows Domäne ist, wird auch die Verbindung zum Domaincontroller überprüft.

5.1 Serverstatus des Domaincontrollers

Um die Verfügbarkeit des Domaincontrollers zu prüfen, wird ein Verbindungstest zu dem Server ausgeführt. Durch diese Methode wird ermittelt, ob der Server prinzipiell verfügbar ist. Der Status des Active Directory kann und wird mit dieser Methode nicht abgefragt.

Vorteilhaft ist, dass diese Methode das Netzwerk nicht stark belastet (verursacht noch weniger Netzwerkverkehr als ein „Ping“) und die Hauptfehlerquelle, nämlich eine Unterbrechung der Netzwerkverbindung, trotzdem erkannt werden kann.

Die Funktion „IsDestinationReachableA” *[5.1]* wird in der Datei „Sensapi.dll“ bereitgestellt und somit auf Windowsbetriebsystemen verfügbar.

Nach folgender Deklaration kann die Funktion im VBA Code von RSView SE verwendet werden:

```
Private Declare Function IsDestinationReachable Lib _
  "Sensapi.dll" Alias "IsDestinationReachableA" _
  (ByVal lpszDestination As String, lpQOCInfo As QOCINFO) _
  As Boolean
```

VBA Code 6: Deklaration der "IsDestinationReachable” Funktion

Die deklarierte Funktion erwartet als Parameter die Zieladresse des zu prüfenden Computers und eine Datenstruktur vom Typ QOCINFO (Quality of Connection Info), in die Daten der Verbindung zurück geschrieben werden.

Die Funktion wird „TRUE“ zurückgeben. wenn zu dem Ziel verbunden werden kann, anderenfalls wird „FALSE“ zurückgegeben.

Die Deklaration der benutzerdefinierten Struktur geschieht folgendermaßen:

```
Private Type QOCINFO
  dwSize As Long
  dwFlags As Long
  dwInSpeed As Long
  dwOutSpeed As Long
End Type
```

VBA Code 7: Deklaration der Quality of Connection Struktur

Nach der Deklaration der Funktion und der Datenstruktur kann die Funktion verwendet werden:

```
Public Function CheckQOC(strServerIP As String) As Boolean
  On Error GoTo Errorhandler
    Dim ConnInfo As QOCINFO

    ConnInfo.dwSize = Len(ConnInfo)

    If IsDestinationReachable(strServerIP, ConnInfo) = 1 Then
      CheckQOC = True
    Else
      CheckQOC = False
    End If

  Exit Function
Errorhandler:
    CheckQOC = False
End Function
```

VBA Code 8: Test der QOC

Das Ergebnis der Funktion „CheckQOC“ wird zur Animation des Serverstatus verwendet. Wenn der Verbindungstest zu der, der Funktion übergebenen IP Adresse (also die IP Adresse der Domaincontroller) erfolgreich war, wird die Statusanzeige auf grün geschaltet. Sollte das Ziel nicht erreichbar sein oder im Funktionsablauf ein Fehler auftreten, wird „FALSE“ zurückgegeben. Die Statusanzeige wird auf rot geschaltet.

5.2 Serverstatus der HMI-, Data- und FactoryTalk Server

Um den Serverstatus abzufragen, wird vom RSView SE Client Object Model eine Funktion zur Verfügung gestellt.
Die Funktion „GetServerStatus“ befindet sich in der Klasse „Appllication“ der Bibliothek „DisplayClient“.

Durch diese Funktion kann der primäre und sekundäre Status eines Servers sowie der Name des aktiven Computers ermittelt werden.

```
GetServerStatus FullServerName,
                PrimaryStatus,
                SecondaryStatus,
                ActiveComputerName
```

VBA Code 9: Test der QOC

Die Werte für den Serverstatus und den aktiven Computernamen werden in die angegebenen Variablen zurück geschrieben. Der Name des zu prüfenden Servers muss mit voller Pfadangabe aus dem RSView Projekt angegeben werden (z.B. „/Building/Line4/Mixer:MixingServer“).

Die Werte der Serverstatus werden als ganzzahlige Werte vom Typ Long zurückgegeben und haben folgende Bedeutungen:

0 = Server aktiv
1 = Server in Standby
2 = Server ist „Out of Service“ also nicht erreichbar
3 = sekundärer Server nicht angelegt

Durch diese Rückgabewerte ist schon eine gute Diagnose der einzelnen Server möglich. Allerdings wird der Fall, dass ein Server abgefragt wird, der in dem Projekt überhaupt nicht angelegt ist, nicht abgefangen.

Zu diesem Zweck wurde die ursprüngliche Funktion in eine neue Funktion eingebettet, die genau diesen Fehlerfall (Fehler -2147220344) abfangen kann.

```
Public Function GetSrvStatus(strServer As String, _
                             lngPriStat As Long, _
                             lngSecStat As Long, _
                             strActiveComputer As String)

  On Error GoTo Errorhandler

    GetServerStatus strServer, _
                    lngPriStat, _
                    lngSecStat, _
                    strActiveComputer

  Exit Function

Errorhandler:
  Select Case Err.Number
   Case -2147220344
    lngPriStat = 4
    lngSecStat = 4
   Case Else
      MsgBox "(" + CStr(Err.Number) + ") " + Err.Description
   End Select
End Function
```

VBA Code 10: Neue Serverstatus Funktion

Die abrufbaren Statusinformationen für einen Server sind jetzt:

0 = Server aktiv

1 = Server in Standby

2 = Server ist „Out of Service“ also nicht erreichbar

3 = sekundärer Server nicht angelegt

4 = Server nicht im Projekt angelegt

Mit diesen Informationen können die entsprechenden Elemente auf dem Display animiert werden.

5.3 Ermittlung der Computernamen für den Serverstatus

Neben dem Status des logischen Servers, also dem Status eines HMI- oder Dataservers, soll auch die Anzeige der physikalischen Computer aus denen ein HMI- oder Dataserver besteht möglich sein.
Bei aktivierter Redundanz besteht ein HMI- bzw. Dataserver aus zwei physikalischen PCs. Eine Auflösung des Servernamens in die Computernamen ist in RSView SE nicht möglich. Aus diesem Grund wurde eine Konfigurationsdatei angelegt, die die benötigten Informationen enthält.
Zum Einlesen der Konfigurationsdatei wurden zwei Vorgehensweisen entwickelt und nach einer Gegenüberstellung der Vor- und Nachteile der Vorgehensweisen die geeignetere Variante ausgewählt und umgesetzt.

5.3.1 Variante 1: Konfiguration als XML Datei

Da bei der Verwendung von VBA unter zur Hilfename der Microsoft Core XML Services (MSXML) auch der Umgang mit XML Dateien möglich ist *[5.2]*, wird bei diesem Konzept eine XML Datei nach dem zu einem logischen Server gehörenden Computer abgefragt.
Zunächst wurde eine XML Datei erstellt, die von der Struktur den Informationen aus dem Projektbaum einer RSView SE Applikation ähnelt.

Hierzu wurden verschiedene XML Elemente verwendet. Das Wurzelelement heißt in Anlehnung an die Struktur eines RSView SE HMI Projektes `HMIProjects`. Eine wohlgeformte XML Datei enthält genau ein Element als Wurzelelement. Des Weiteren gibt es die Elemente `HMIProject` (beschreibt die Struktur eines Projektes, auf einem Server können mehrere Projekte angelegt sein), `Area` und `Server`.
Beim Aufbau der XML Datei sind folgende Regeln einzuhalten:

1. Es darf nur ein Element vom Typ `HMIProjects` geben (Wurzelelement)
2. Das Element `HMIProjects` darf beliebig viele Elemente `HMIProject` enthalten aber, kein Element vom Typ `Area` oder `Server`

3. Ein Element `HMIProject` darf beliebig viele Elemente vom Typ `Area` und `Server` enthalten und muss das Attribut `Name` besitzen
4. Ein Element `Area` darf beliebig viele Elemente vom Typ `Server` enthalten und muss das Attribut `Name` besitzen
5. Ein Element `Server` muss die Elemente `PrimaryComputer` und `SecondaryComputer` enthalten und die Attribute `Name` und `Servertyp` besitzen.

Durch diese Regeln lassen sich fast beliebige Strukturen eines oder mehrerer RSView Projekte auf einem Server nachbilden.

Im Gegensatz zu den Möglichkeiten, die die XML Datei bei der Abbildung der Projektstruktur bietet, sind bei einem realen Projekt aber folgende Einschränkungen vorhanden:

1. Maximal ein HMI Server pro Area (aber mehrere Dataserver möglich)
2. Maximal ein Server pro Area (empfohlen)
3. Maximal zehn HMI Server insgesamt

Eine nach diesen Regeln aufgebaute XML Datei hat folgende Struktur:

```
<?xml version="1.0" encoding="UTF-8"?>
<HMIProjects>
  <HMIProject Name="NCV3">
    <Area Name="06511">
      <Server Name="06511_A1" ServerTyp="HMI">
        <PrimaryComputer>1100000hmi010</PrimaryComputer>
        <SecondaryComputer>8000000hmi020</SecondaryComputer>
      </Server>
      <Area Name="DA1">
        <Server Name="11OFS010" ServerTyp="DS">
          <PrimaryComputer>1100000OFS010</PrimaryComputer>
          <SecondaryComputer>1100000OFS050</SecondaryComputer>
        </Server>
      </Area>
     +<Area Name="DA2">
     +<Area Name="DA3">
     +<Area Name="DA4">
    </Area>
   +<Area Name="06512">
  </HMIProject>
</HMIProjects>
```

Abb. 16: RSView Projekt Struktur in XML Datei

Die gewünschten Informationen sind jetzt in der XML Datei enthalten und müssen aus VBA heraus abgefragt werden.

Unter zur Hilfename des MSXML wird zunächst eine Verbindung zu der XML Datei aufgebaut. Um auch hier Redundanz zu gewährleisten, wird nach einem Fehlversuch ein Verbindungsaufbau zu einer zweiten Quelle durchgeführt. Die XML Dateien können dabei in einer Netzwerkfreigabe auf einem anderen Computer auf der lokalen Festplatte oder auf einem Webserver liegen.
In diesem Fall bietet sich die Bereitstellung über einen Webserver an, da alle Rockwell HMI Server in einer verteilten Umgebung auch über einen Webserver verfügen.

Zunächst werden die benötigten Variablen deklariert:

```
Dim objXMLDocument As New MSXML2.DOMDocument40
Dim objXMLDocumentElement As Object
```

VBA Code 11: XML Abfrage - Deklaration der Variablen

Nun kann die Verbindung zur ersten Quelle aufgebaut werden. Sollte dieser Versuch fehlschlagen, wird automatisch zu einer zweiten Quelle verbunden. Sollte auch dieser Versuch fehlschlagen, wird die Abfrage abgebrochen.

```
strXMLPath1 = "http://server1/xmlconfig1.xml
strXMLPath2 = "http//server2/xmlconfig2.xml"

If objXMLDocument.Load(strXMLPath1) = False Then

  If objXMLDocument.Load(strXMLPath2) = False Then

    Debug.Print objXMLDocument.parseError.reason
    Set objXMLDocument = Nothing
    Exit Sub

  End If

End If

Set objXMLDocumentElement = objXMLDocument.documentElement
```

VBA Code 12: XML Abfrage - Verbinden zur Datenquelle

Nachdem die Verbindung zu einer XML Datei hergestellt wurde, kann eine Abfrage mittels XPath (XML Path Language) erfolgen.
XPath ist eine vom W3C-Konsotium entwickelte Abfragesprache, um bestimmte Teile in einer XML Datei zu erreichen.
Bei dieser Art der Abfrage wird der Pfad zu dem gewünschten Element angegeben.
Der Ausdruck „`/HMIProjects/HMIProject[@Name='NCV3']`" wählt z.B. das Element `HMIProject` aus der XML Datei aus, dass das Attribut `Name` mit dem Inhalt „NCV3" enthält.

Um mit einem Ausdruck alle Server aus einem Projekt abfragen zu können, wird der Pfad zum Teil durch Variablen aufgebaut.

```
strServerName = "06511_A1" 'Servername nach dem gesucht wird

strPriComputerName = objXMLDocumentElement.selectSingleNode _
  ("/HMIProjects
    /HMIProject[@Name='NCV3']
    //Server[@Name='" + strServerName + "']
    /SecondaryComputer").Text

strSecComputerName = objXMLDocumentElement.selectSingleNode _
  ("/HMIProjects
    /HMIProject[@Name='NCV3']
    //Server[@Name='" + strServerName + "']
    /SecondaryComputer").Text
```

VBA Code 13: XML Abfrage - XPath Abfrage

In diesem Fall wird das Projekt „NCV3" ausgewählt und danach der Server mit dem angegeben Namen ausgewählt. Eventuell vorhandene Areas werden übersprungen (an „//" zu erkennen).
Zurückgegeben wird der Text des Elements `PrimaryComputer` bzw. bei der zweiten Abfrage der Text des Elements `SecondaryComputer`.

Nach der Abfrage der gewünschten Daten kann die Verbindung einfach durch Aufheben der zu Begin deklarierten Objekte beendet werden.

```
Set objXMLDocument = Nothing
Set objXMLDocumentElement = Nothing
```

VBA Code 14: XML Abfrage - Verbindung beenden

5.3.2 Variante 2: Konfiguration als RSView Parameter Datei

Da das Hauptmerkmal der Konfigurationsdatei die Auflösung der Servernamen in Computernamen ist, wurde unter zur Hilfename der RSView SE Parameterdateien eine Variante entwickelt, mit der genau diese Funktion umgesetzt wird.
Bei einer Parameterdatei handelt es sich im Prinzip um eine Liste einzelner Zeichenketten, die einfach von #1 mit #n durchnummeriert sind.
Der ursprüngliche Zweck einer Parameterdatei ist es, eine ganze Reihe an Parametern einem Display beim Aufruf mit zu übergeben. Genauer gesagt werden durch eine Parameterdatei eine ganze Reihe an „Parametertags" übergeben.

Bei einem Display in RSView können einzelne Parametertags übergeben werden. So hat z.B. der Befehl

```
Display statusbild1 /TErsterParameter
```

zur Folge, dass in dem aufgerufenen Bild alle enthaltenen Platzhalter `#1` (weil erster Parameter) durch `ErsterParameter` ersetzt werden.

Es ist auch möglich, mehrere Parameter zu übergeben. Der Befehl

```
Display statusbild1 /TErsterParameter, ZweiterParameter
```

hat zur Folge, dass im aufgerufenen Bild alle enthaltenen Platzhalter `#1` durch `ErsterParameter` und alle enthaltenen Platzhalter `#2` durch `ZweiterParameter` ersetzt werden.

Im Prinzip können auf diese Art und Weise beliebig viele Informationen an ein Display übergeben werden – allerdings ist es leicht zu erkennen, dass der Umgang mit einer großen Anzahl an Parameter schnell unübersichtlich wird. Aus diesem Grund ist es in einem RSView SE Projekt möglich, Dateien mit Parametern anzulegen, zu speichern und die ganzen Dateien einem Display zu übergeben.

Eine angelegt Parameterdatei hat folgende Struktur:

```
#1   = ErsterParameter
#2   = ZweiterParameter
...
#n   = n-ter Parameter
```

Abb. 17: Prinzip einer Parameterdatei

Wird jetzt der Befehl

```
Display statusbild /PNameDerParameterdatei
```

ausgeführt, werden die Platzhalter #1 bis #n in den Ausdrücken des Displays ersetzt.

Da es jetzt möglich ist mit einem einfachen Displayaufruf eine große Menge an Informationen zu übergeben, wurde eine Parameterdatei erstellt, in der alle benötigten Informationen zur Ermittlung der zu einem Server gehörenden Computernamen enthalten sind.
Dabei wurde folgende Struktur festgelegt:

```
#Eintrag A     =(Full)Servername
#Eintrag A + 1 = primärer Computername
#Eintrag A + 2 = sekundärer Computername
#Eintrag A + 3 = Servertyp (HMI, DS, FT)
```

Abb. 18: Parameterdatei zur Ermittlung der Computernamen

Sollte ein Server keinen sekundären Computer haben (also keine Redundanz besitzen) so muss der entsprechende Platz in der Parameterdatei freigelassen werden.

```
#Eintrag B     =(Full)Servername
#Eintrag B + 1 = primärer Computername
#Eintrag B + 2 =
#Eintrag B + 3 = Servertyp (HMI, DS, FT)
```

Abb. 19: Parameterdatei zur Ermittlung der Computernamen (ohne Redundanz)

Da die übergebenen Daten nicht in einem Ausdruck auf dem Display verwendet werden, es also keine Platzhalter (#1 - #n) auf dem Display gibt, müssen die übergebenen Daten in dem VBA Programm gelesen und weiter verarbeitet werden. Nach dem Displayaufruf sind alle übergebenen Parametertags in einer „Stringlist Collection“ vorhanden. Bei diesen Daten handelt es sich um eine Sammlung aneinandergereihter Zeichenketten. Zu der Sammlung können im Programmablauf keine Daten mehr hinzugefügt werden. Eine Sammlung dieser Art kann auch nicht im Programmablauf erstellt werden.

Ein Lesezugriff ist aber möglich und wird durch die Methode „Item“ (Zugriff auf ein bestimmtes Element) und die Eigenschaft „Count“ (Anzahl der eingetragenen Elemente) erleichtert.

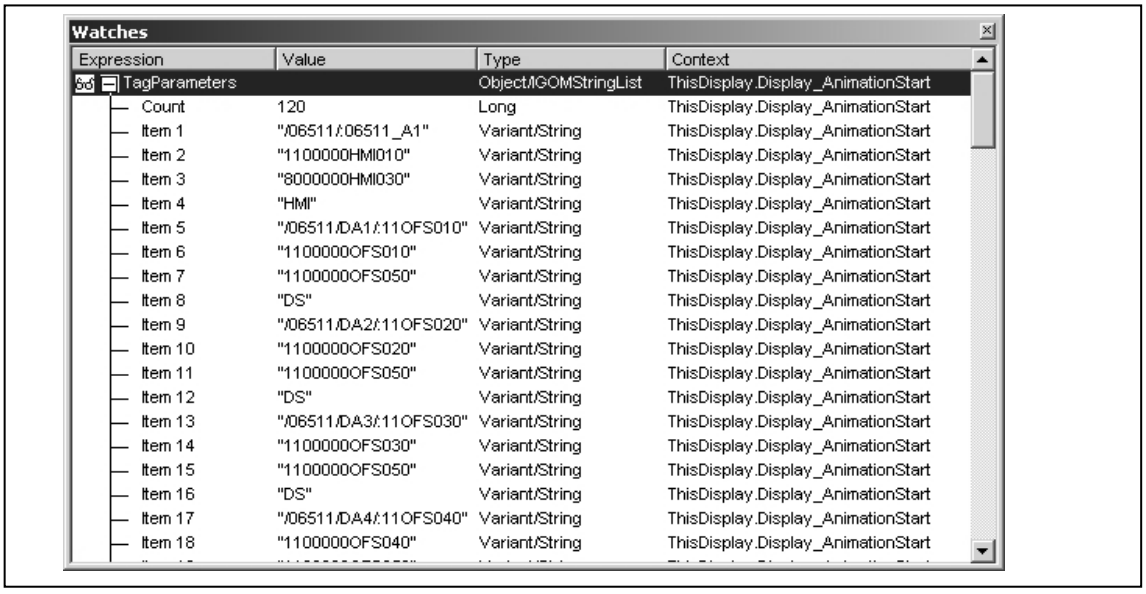


Abb. 20: Überwachungsfenster mit geladenen Parametertags

Um einen bestimmten Servernamen in Computernamen aufzulösen, wird die Sammlung `Tagparameter` vom ersten Element bis zum letzten Element nach dem gesuchten Servernamen durchsucht. Wenn der Servername gefunden wurde, enthalten die nächtens drei Elemente die gewünschten Informationen über primären und sekundären Computer sowie den Servertyp.

Um den Umgang mit der benötigten Parameterdatei für RSView SE zu erleichtern, wurde eine Arbeitsmappe in Microsoft Excel angelegt. In dieser Arbeitsmappe kann die Konfiguration der Server vorgenommen werden und

nach Fertigstellung in eine Parameterdatei für RSView exportiert werden. Ein Bearbeiten einer vorhandenen Parameterdatei ist mit dem Exceltool allerdings nicht möglich. Es können nur neue Dateien erstellt werden. Die Informationen bleiben jedoch in der Exceldatei erhalten und können immer wieder verwendet werden.

Full Server Name			Primary Computer	Secondary Computer	Servertyp	Cell	Description
Area	Subarea	Servername					
/06511		/:06511_A1	1100000HMI010	8000000HMI030	HMI		
/06511	/DA1	/:11OFS010	1100000OFS010	1100000OFS050	DS		
/06511	/DA2	/:11OFS020	1100000OFS020	1100000OFS050	DS	11	Rahmen
/06511	/DA3	/:11OFS030	1100000OFS030	1100000OFS050	DS		
/06511	/DA4	/:11OFS040	1100000OFS040	1100000OFS050	DS		
/06512		/:06512_A1	1200000HMI010	8000000HMI030	HMI		
/06512	/DA1	/:12OFS010	1200000OFS010	1200000OFS040	DS	12	Unterbau
/06512	/DA2	/:12OFS020	1200000OFS020	1200000OFS040	DS		
/06512	/DA3	/:12OFS030	1200000OFS030	1200000OFS040	DS		
/06513		/:06513_A1	1300000HMI010	8000000HMI040	HMI		
/06513	/DA1	/:13OFS010	1300000OFS010	1300000OFS050	DS		
/06513	/DA2	/:13OFS020	1300000OFS020	1300000OFS050	DS	13	Fördertechnik
/06513	/DA3	/:13OFS030	1300000OFS030	1300000OFS050	DS		
/06513	/DA4	/:13OFS040	1300000OFS040	1300000OFS050	DS		
/06514		/:06514_A1	1400000HMI010	8000000HMI040	HMI		
/06514	/DA1	/:14OFS010	1400000OFS010	1400000OFS060	DS		
/06514	/DA2	/:14OFS020	1400000OFS020	1400000OFS060	DS	14	Seitenwand
/06514	/DA3	/:14OFS030	1400000OFS030	1400000OFS060	DS		
/06514	/DA4	/:14OFS040	1400000OFS040	1400000OFS070	DS		
/06514	/DA5	/:14OFS050	1400000OFS050	1400000OFS070	DS		
/06515		/:06515_A1	1500000HMI010		HMI	15	Musterprojekt
/06515	/DA1	/:15OFS010	1500000OFS010	1500000OFS020	DS		

Abb. 21: Struktur der Exceltabelle zum Erstellen von Parameterdateien

Falls es gewünscht ist, kann die erstellte Parameterdatei auch direkt im RSView Studio oder, falls kein RSView Studio vorhanden, kann die Datei auch mit einem einfachen Texteditor editiert werden.

5.3.3 Auswahl der geeigneteren Variante

Vorteile Variante 1 (XMLDatei):

1. hoher Informationsgehalt der Datei (die Struktur des gesamten Projektes wird abgebildet)
2. strukturierte Konfigurationsdatei
3. gezielte Abfragen möglich

Nachteile Variante 1 (XMLDatei):

1. Konfigurationsdatei befindet sich nicht in dem RSView Projekt

Vorteile Variante 2 (Parameterdatei):

1. einfache Struktur
2. Konfigurationsdatei befindet sich innerhalb des RSView Projekts

Nachteile Variante 2 (Parameterdatei):

1. Reihenfolge der angegebenen Dateien muss zwingend eingehalten werden
2. bei großen Datenmengen unübersichtlich
3. keine gezielte Datenabfrage möglich

Trotz der Vorteile der Variante 1 (XML Datei) wie komfortabler Zugriff auf die Daten und eine übersichtliche Darstellung der Daten wiegt der Nachteil, dass die Konfigurationsdatei außerhalb des RSView Projekts liegt so schwer, dass die andere Variante gewählt wurde. Der kompliziertere Umgang mit der Parameterdatei wurde zum Teil durch das erstellte Exceltool ausgeglichen. Der Hauptvorteil ist jedoch, dass sich die Konfigurationsdatei innerhalb des RSView Projektes befindet und so z.B. beim Replizieren mit dem sekundären Server automatisch mitgesichert wird. Eine Extrabehandlung dieser Datei ist nicht nötig.

6 FactoryTalk Event Viewer als RSView SE Bild

Ein weiteres wichtiges Diagnoseinstrument ist das FactoryTalk Eventlog. Hierbei handelt es sich um ein Ereignisprotokoll, in das FactoryTalk und FactoryTalk basierende Anwendungen Nachrichten aller Art speichern können. Bei der Installation von FactoryTalk wird automatisch ein neues Windows Eventlog (auf deutschen Systemen „Ereignisanzeige“ genannt) auf dem lokalen System angelegt, in das zukünftig die entsprechenden Nachrichten geschrieben werden.
Der Vorteil dieses Vorgehens liegt darin, dass Informationen zentral gesammelt werden. Da keine programmspezifischen Dateien benutzt werden, entfallen das Suchen nach den Logdateien und eventuelle Probleme beim Auslesen der Dateien.

Nachrichten, die in das FactoryTalk Eventlog geschrieben werden, sind in der „Diagnostic List“ der entsprechenden Anwendungen zu sehen. Neben der aktuellen Meldung, die an erster Stelle der Liste steht, ist es auch möglich, vergangene Meldungen anzuzeigen in dem nach unten geblättert wird.

Abb. 22: RSView Studio Diagnostic List

Die Meldungen betreffen alle möglichen Arten von Informationen die während des Programmablaufs anfallen können wie

- Client Startvorgang beenden
- Server erreichbar/ nicht erreichbar
- Ausgeführte Kommandos (z.B. Displayaufruf)
- Starten / Stoppen des Alarmings oder Datenlogs
- Informationen über Tags die nicht gelesen werden konnten
- usw.

Dank dieser vielfältigen Informationen ist es möglich, Probleme auch im Nachhinein nachvollziehen zu können. Diese Möglichkeiten werden durch eine Unterscheidung der verschiedenen Nachrichtentypen noch verbessert. So sind die anfallenden Nachrichten in vier Kategorien gegliedert:

Error:	Hinweise auf kritische Probleme
Warning:	Hinweise auf im Moment nicht kritische Probleme
Info:	Normale Informationen
Audit:	Meldungen über Veränderungen an der Anwendung z.B. Bearbeitung eines RSView Displays

Da die Diagnoseleiste einer Anwendung jedoch ausgeblendet werden kann oder es notwendig sein kann, die Informationen auch nach dem Beenden der Anwendung einsehen zu können, wird mit FactoryTalk zusammen ein Programm zum Auslesen des Eventlogs, ein so genannter „Event Viewer", mitinstalliert.

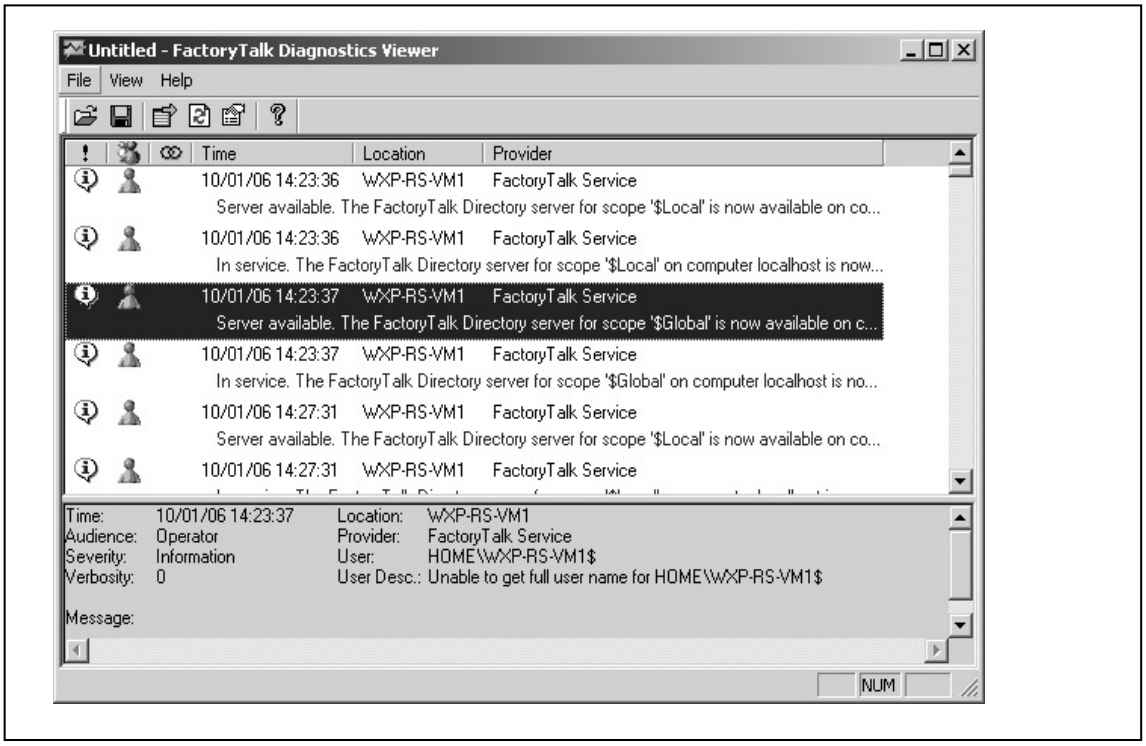


Abb. 23: FactoryTalk Diagnostics Viewer

Mit diesem Programm ist es möglich, Nachrichten aus der Vergangenheit anzeigen zu lassen. Außerdem stehen einige Filterfunktionen zur Verfügung, mit

denen - z.B. nur auf Fehlermeldungen (Typ „Error") - gefiltert werden kann. Als Informationsquelle dient diesem Programm das Windows Event Log.
Eine weitere Möglichkeit die gesammelten Informationen auszulesen, ist das in Windows integrierte Snap-In „Microsoft Event Viewer" für die Microsoft Management Console (MMC). Mit diesem Programm kann nicht nur auf das FactoryTalk Eventlog zugegriffen werden, sondern auch auf die drei standardmäßig vorhandenen Eventlogs:

- **Anwendungsprotokoll**
  Ereignisse von einzelnen Anwendungen. Welche Ereignisse überwacht und gespeichert werden, entscheidet der Entwickler des Programms
- **Sicherheitsprotokoll**
  Enthält Ereignisse wie gültige und ungültige Anmeldeversuche. Welche Ereignisse im Einzelnen protokolliert werden, wird von dem Systemadministrator festgelegt
- **Systemprotokoll**
  Das Systemprotokoll enthält Ereignisse, die von Systemkomponenten aufgezeichnet werden. Die Ereignisse können z.B. das Laden eines Treibers oder auch den Start von Diensten betreffen

Da Ereignisse, die im FactoryTalk Eventlog gespeichert werden, auch mit anderen Ereignissen auf dem System zusammen hängen können (z.B. FactoryTalk meldet, dass ein Server nicht erreichbar ist – gleichzeitig konnte ein Dienst nicht gestartet werden) hat der Microsoft Event Viewer den großen Vorteil, dass auch direkt auf die anderen Protokolle zugegriffen werden kann.

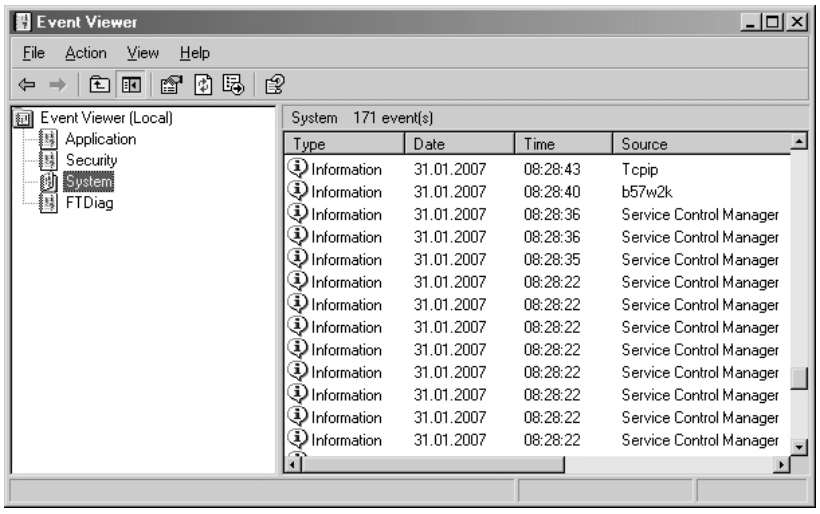


Abb. 24: Microsoft Event Viewer

Wie in der Abbildung zu erkennen ist, sind im Gegensatz zum FactoryTalk Diagnostics Viewer im linken Teil des Fensters alle auf dem Computer verfügbaren Eventlogs zu sehen. In diesem Fall sind es die dreiangesprochenen Standardlogs und das FactoryTalk Log.
Da es, wie schon angesprochen, vorkommen kann, dass mehrere Ereignisse miteinander in Zusammenhang stehen, kann es notwendig sein, mehrere Eventlogs zu durchsuchen. Da bei einer verteilten Architektur allerdings mehrere Computer zum Einsatz kommen, muss man z.B. das FactoryTalk Eventlog auf einem Client überprüfen und aufgrund der gefundenen Meldungen, zusätzlich das Eventlog auf einem der Server.

Auch diese Möglichkeit - die direkte Verbindung zu einem anderen Computer - bietet der Microsoft Event Viewer.

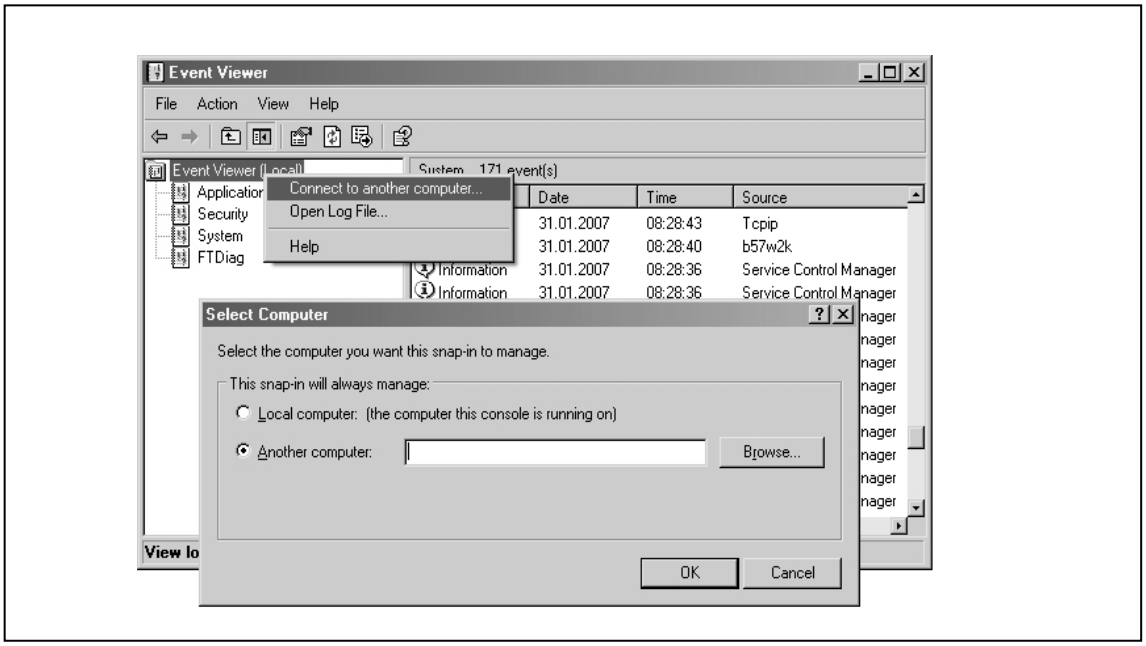


Abb. 25: Microsoft Event Viewer – Verbindung zu einem anderen Computer

Der Zugang zu den verschiedenen Event Viewern ist allerdings nur möglich, wenn der RSView SE Client verlassen werden darf bzw. es erlaubt ist, andere Programme auf dem Client Computer zu starten. Dies wird jedoch aus Sicherheitsgründen oft verboten. So werden die Informationen zwar vom System gesammelt, aber der Zugang zu den Informationen auf dem lokalen Computer ist nicht möglich.

Aus diesem Grund wurde die Möglichkeit untersucht, die Informationen aus dem FactoryTalk Log in einem RSView SE Bild darzustellen. Folgende Gründe sprechen für diese Möglichkeiten:

- Das Display und das eventuell integrierte VBA Programm werden zwar vom HMI Server geladen, aber lokal ausgeführt
- Alle nötigen Elemente zur Darstellung wie Listboxen und Textfelder sind auch in RSView SE Displays verfügbar und können durch ein VBA Programm gesteuert werden
- Der Zugriff auf Objekte des Betriebsystems ist mit der „Microsoft WMI Scripting Library“ möglich, die auch in VBA verwendet werden kann

Durch diese technischen Möglichkeiten kann ein Objekt vom Betriebssystem abgerufen und verarbeitet werden. Um eine gezielte Abfrage zu ermöglichen, wurde die Abfragesprache WQL (Windows Management Instrumentation Query Language) verwendet. Bei dieser Abfragesprache handelt es sich um einen Dialekt des bekannten SQL (Structured Query Language) wie es auch zum Abfragen von Datenbanken wie Microsoft SQL oder MySQL verwendet wird. Die Möglichkeiten dieser Abfragesprache sind jedoch im Gegensatz zu z.B. SQL vom Microsoft SQL Server beschränkt, bieten aber dennoch genügend Möglichkeiten, um gezielt Informationen vom Betriebsystem wie z.B. ein Eventlog abzurufen. So können nur `SELECT` Statements (also Abfragen um Daten abzurufen) ausgeführt werden. Möglichkeiten der Bearbeitung wie `INSERT`, `UPDATE` oder `ALTER` , wie sie aus dem Umgang mit Datenbanken bekannt sind, sind nicht möglich *[6.1]*.
Da die benötigten Daten für einen Event Viewer „nur“ angezeigt werden und es dem Benutzer nicht möglich sein soll Einträge zu verändern, sind die Möglichkeiten eines `SELECT` Statements ausreichend und übertreffen durch die Möglichkeiten der Einschränkungen bei der Abfrage sogar die Filterfunktionen der vorgestellten Event Viewer.

Um Daten abfragen zu können, muss zunächst mit dem lokalen „Windows Management Instrumentation“ verbunden werden:

```
Set objWMIService = GetObject("winmgmts:")
```

VBA Code 15: Verbindung mit lokalem WMI

Durch dieses Vorgehen wird der Standardnamensraum des Computers ausgewählt. Sollten die Standardeinstellungen nicht geändert worden sein, so ist dies „root/cimv2“ (Computer Information Model). In diesem Namensraum liegen die Informationen über den lokalen Computer, unter denen sich auch das Eventlog befindet. Nach dem erfolgreichen Verbindungsaufbau kann eine Abfrage gestartet werden.

```
SELECT * FROM Win32_NTLogEvent WHERE Logfile ='FTDiag'
```

Abb. 26: WQL Abfrage - Alle Einträge aus dem Eventlog "FTDiag"

Durch diese Abfrage werden alle Einträge (**SELECT** * ...) aus dem Windows Eventlog (... **FROM** Win32_NTLogEvent) abgerufen, die sich im Eventlog „FTDiag“ (...**WHERE** Logfile='FTDiag') befinden. Nach dem Ausführen der Abfrage stehen die einzelnen Einträge in einem so genannten SWbemObjectSet (Web Based Enterprise Management Scripting Sammlung) zur Verfügung.

```
Set objColItems = objWMIService.ExecQuery( _
 "SELECT * FROM Win32_NTLogEvent WHERE Logfile ='FTDiag')
```

VBA Code 16: WQL Abfrage - Ausführen einer Abfrage

Da auch mit WQL Abfragen mehrere Bedingungen in der WHERE-Klausel miteinander verknüpft werden können, wurden zunächst alle verfügbaren Felder ausgelesen.

Insgesamt existieren pro Eintrag 16 Felder, auf die sich einzeln filtern lässt:

1. Category
2. CategoryString
3. ComputerName
4. Data
5. EventCode
6. EventIdentifier
7. EventType
8. InsertionStrings
9. Logfile
10. Message
11. RecordNumber
12. SourceName
13. TimeGenerated
14. TimeWritten
15. Type
16. User

Das Feld „Logfile“ wird in der Abfrage schon verwendet, da hier angegeben ist zu welchen der vier Eventlogs dieser Eintrag gehört. Möglich wäre an dieser Stelle auch auf z.B. Meldungen des Typs „Error“ zu Filtern oder sich alle Meldungen aus einer Quelle anzeigen zu lassen.

Eine typische FactoryTalk Meldung enthält folgende Informationen:

Feld	**Inhalt** (beispielhaft)
Date:	*02.02.2007*
Time:	*13:51:21*
Type:	*Information*
User:	*N/A*
Computer:	*W2K3SIDEN-VM-8S*
Source:	*FactoryTalk Service*
Category:	*Operator*
Event ID:	*1001*
Description:	*Logged Date : 2:51:19 Friday, February 02, 2007* *Location: W2K3SIDEN-VM-8S* *Provider: FactoryTalk Service* *Username : WORKGROUP\W2K3SIDEN-VM-8S$* *Verbosity: 0* *In service. The FactoryTalk Directory server for scope '$Global' on computer localhost is now active.*

Abb. 27: Felder eines Eventlog Eintrages

An dieser Stelle ist jedoch ein Nachteil der FactoryTalk Meldungen zu erkennen. Es ist zwar ein Feld „Event ID“ (also eine Identifikationsnummer für eine bestimmte Art von Ereignissen) vorhanden, wird jedoch nicht richtig genutzt. Alle von FactoryTalk angelegten Einträge erhalten die Event ID 1001, so dass eine weiter Auswertung an dieser Stelle nicht möglich. Des Weiteren wird in das vorgesehene Feld „User“ kein Benutzer eingetragen, so dass auch diese Informationen nicht zur Verfügung stehen.

Bei den FactoryTalk Meldungen sind alle relevanten Informationen in dem Text der Nachricht vorhanden. Durch dieses Vorgehen kann mit den schon vorgestellten Event Viewern (FactoryTalk Diagnostics Viewer und Microsoft Eventviewer) nicht auf bestimmte Ereignisse gefiltert werden, die sich nur durch bestimmte Schlüsselwörter in dem Text auszeichnen. Sollte ein Server wieder verfügbar sein, so wird eine Meldung vom Typ „Information“ angelegt, kann jedoch ansonsten nur durch die Schlüsselwörter „In service …“ erkannt werden. Ist jetzt eine Liste aller Meldungen gewünscht, die die Widerverfügbarkeit von Servern betrifft, ist dies mit den klassischen Eventviewern nicht möglich. Dank der WQL Abfragen ist es jedoch möglich, auch auf Meldungen zu filtern die bestimmte Schlüsselwörter enthalten. Hierzu wird wieder eine WQL Abfrage gestartet, die in der WHERE- Klausel nicht nur das entspreche Eventlog enthält (...**WHERE** Logfile='FTDiag'), sondern auch ein LIKE Operator. Denkbar wäre auch eine Abfrage, in die einfach eine Bedingung eingefügt wird, dass „Message“ die Zeichenkette „In service“ sein soll.

```
SELECT * FROM Win32_NTLogEvent
WHERE Logfile ='FTDiag'
AND Message = 'In service'
```

Abb. 28: WQL Abfrage - Filter auf „In service“ ohne LIKE

Allerdings wird diese Abfrage zu keinem Ergebnis führen, da es keine Meldungen gibt, die nur den Text „In service“ beinhalten. Dies wird jedoch durch das Gleichheitszeichen abgefragt (also der Text muss exakt „In Service“ sein). Anders verhält es sich mit dem LIKE Operator. Hier wird überprüft, ob das

angegebene Feld die angeforderte Zeichenkette enthält. Außerdem kann schon während der Abfrage angegeben werden, dass sich z.B. vor oder nach dem Kriterium eine unbekannte Anzahl von unbekannten Zeichen befindet.
Eine Abfrage unter zur Hilfename des LIKE Operators, um alle Meldungen aus dem Eventlog zu Filtern, die eine Wiederverfügbarkeit eines Servers betreffen sieht dann folgendermaßen aus:

```
SELECT * FROM Win32_NTLogEvent
WHERE Logfile ='FTDiag'
AND Message LIKE 'In service%'
```

Abb. 29: WQL Abfrage - Filter auf „In service“ mit LIKE

Nun werden alle Meldungen angezeigt, die die angegebene Zeichenkette und eine unbekannte Anzahl unbekannter Zeichen enthalten. Durch dieses Vorgehen ist es möglich, die vier wichtigsten Nachrichten herauszufiltern, die den Status der Server betreffen:

1. **Server available**
   Typ: Information
   Bedeutung: Server erreichbar aber nicht aktiv
2. **In service**
   Typ: Information
   Bedeutung: Server ab jetzt aktiv
3. **Server unavailable**
   Typ: Warnung
   Bedeutung: Die Verbindung zu dem auf dem Server laufenden Dienst wurde unterbrochen
4. **Out of service**
   Typ: Fehler
   Bedeutung: Der angegebene Dienst ist auf keinem Server verfügbar

Das erstellte RSView SE Displays sollte von der Art der dargestellten Informationen dem „Microsoft Event Viewer“ ähneln, da sich diese Darstellung bewährt hat.

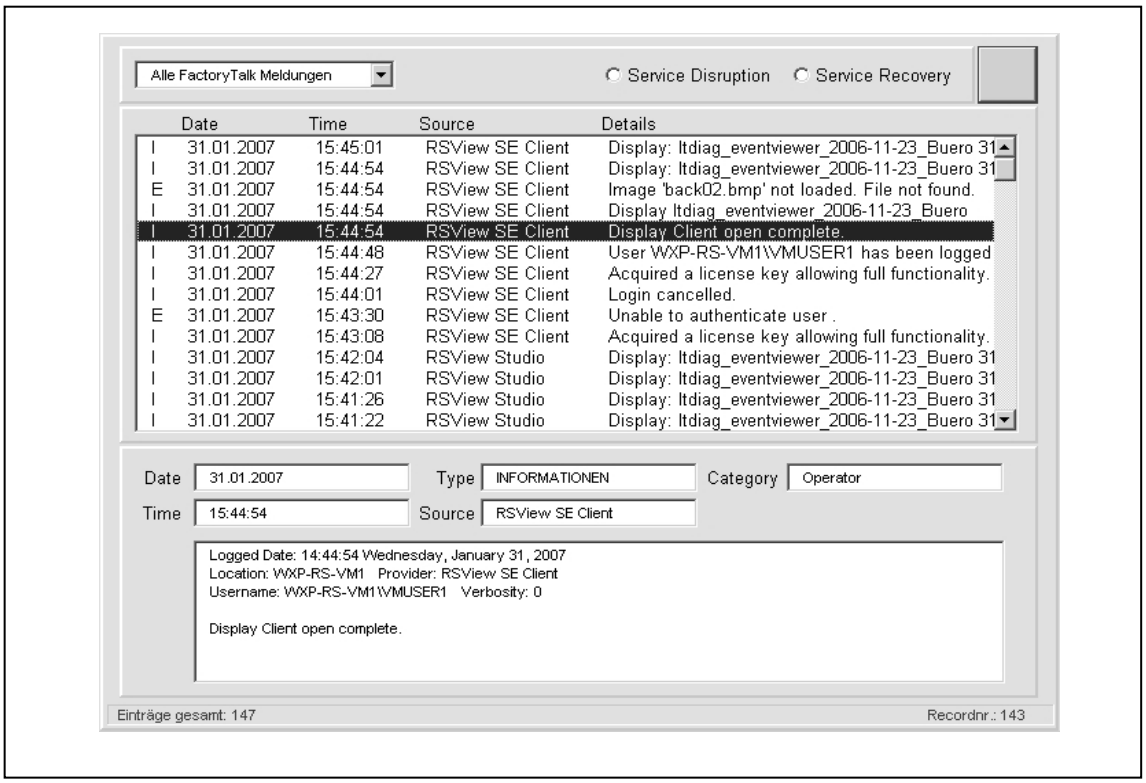


Abb. 30: Event Viewer als RSView SE Display

6.1 Einschränkungen Event Viewers

Nach der Entwicklung und dem Test des Bildes in einer Entwicklungsumgebung wurde das Bild in dem Zielsystem installiert. Beim Benutzen des Bildes sind Probleme durch die große Menge an gespeicherten Meldungen aufgetreten. Auf Grund der eingestellten maximalen Größe des FactoryTalk Event Logs können sich ca. 25000 Meldungen in einer Logdatei ansammeln. Diese Menge an Informationen hat zur Folge, dass die Abfrage sehr lange Zeit in Anspruch nimmt und das Display während dieser Zeit nicht reagiert. Die Abfrage funktioniert während dieser Zeit jedoch stabil, und die Ergebnisse werden nach erfolgreicher Abfrage korrekt angezeigt. Nach Auftreten dieser Probleme wurde das Programm des Displays angepasst, so dass die vorgenommenen Eingaben (z.B. Beenden des Displays) abgearbeitet werden können. Durch diese Maßnahme ist die gesamte Verarbeitungszeit noch weiter gestiegen. Da es den Benutzern nicht zugemutet werden kann, mehrer Minuten auf das Ergebnis zu warten, wurden verschiedene Möglichkeiten untersucht, um die Abfrage einzuschränken, damit nicht alle ca. 25000 Einträge verarbeitet werden.

1. Limitierung der zu verarbeitenden Zeilen mit `SELECT TOP …`. Diese Einschränkung würde das Ergebnis auf eine angegebene Anzahl von Zeilen beschränken
2. Einschränkung des Zeitraums aus dem die Meldungen sein dürfen (z.B. nur Meldungen aus den letzten 48 Stunden)
3. Zählen der Meldungen mit `SELECT Count(*)` und gezielte Abfrage eines bestimmten Bereichs von Meldungen (z.B. RecordNumber 1000 – 2000)

Bei den Untersuchungen sind die Einschränkungen von WQL gegenüber normalen SQL Abfragen deutlich zum Tragen gekommen. So ist ein Zählen der Einträge mittels `Count(*)` in WQL genauso wenig möglich wie eine Einschränkung der abzufragenden Zeilen. Diese Möglichkeiten bietet die Abfragesprache EWQL (Extended WQL) welche aber auf den eingesetzten Windows XP bzw. Windows Server 2003 Betriebsystemen nicht zur Verfügung

steht. EWQL unterstützt weitere SELECT- Klauseln wie DISTINCT, COUNT, JOIN, ORDER BY. Sie ist aber nur im Rahmen eines Microsoft System Management Servers (SMS) verfügbar. Eine Einschränkung des Zeitraums liefert unregelmäßige Ergebnisse, da zur Abfrage eine Konvertierung des Zeitformat vorgenommen werden muss, die, trotzt speziell für diesen Zweck vorgesehenen Funktionen, nur unzuverlässig funktioniert. Bei einer eventuellen Weiterentwicklung dieses Eventviewers bietet diese Lösung die meisten Aussichten auf Erfolg.

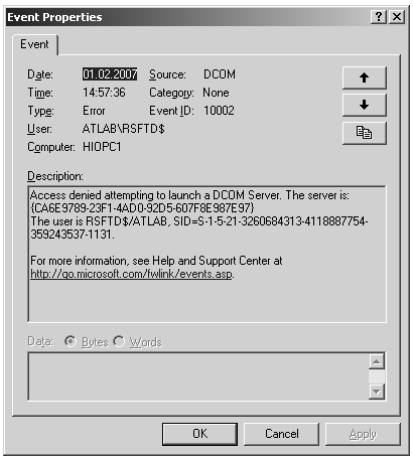
Aufgrund dieser Probleme kann der Eventviewer als RSView SE Bild trotzt überlegener Möglichkeiten der Filterungen und einer frei gestaltbaren Oberfläche nicht eingesetzt werden.

7 Anwendungsbeispiel: Eventlog zur Fehlerdiagnose

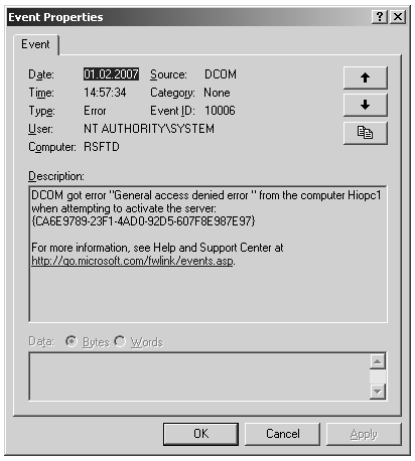
Bei der Inbetriebnahme eines OPC Servers der Firma Hirschmann wurden die aufgetretenen Fehler durch systematische Analyse der Ereignisprotokolle der beteiligten Systeme erkannt und konnten so gezielt behoben werden. Dieses Beispiel wird an dieser Stelle aufgeführt, da es deutlich zeigt, welch ein leistungsfähiges Instrument zur Fehlerbehebung mit einem ausführlichen Eventlog zur Verfügung gestellt wird.

7.1 Ausgangssituation

Auf einem allein stehenden Computer wurde der OPC Server „HiOPC“ der Firma Hirschmann installiert, um Gerätedaten aus einem Switch über den OPC Server in der Visualisierung bereitstellen zu können. Bei ersten Zugriffen auf den OPC Server traten keine Problem auf und die Daten konnten lokal mittels einem OPC Clients gelesen werden. Nachdem der OPC Server in die Visualisierung aufgenommen wurde, konnten keine Daten in RSView gelesen werden. Bei Betrachtung der gestarteten Prozesse auf dem Computer des HiOPC viel auf, dass der OPC Enumerator zwar gestartet war, der Prozess des HiOPC (Industrial SNMP OPC Server der Firma COI Software) selber aber nicht lief. Bei Betrachtung des Eventlogs auf dem Client, des HiOPC Server und des FactoryTalk Servers wurden folgende Einträge gefunden:



Eventlog Eintrag auf dem HiOPC Server



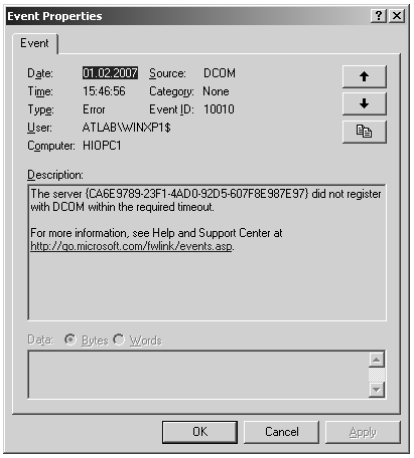
Eventlog Eintrag auf dem FactoryTalk Server

Abb. 31: DCOM Fehler - Start nicht möglich

Die Meldungen deuten darauf hin, dass die entsprechenden Prozesse auf dem HiOPC von FactoryTalk nicht gestartet werden konnten.

7.2 Geänderte DCOM Startberechtigungen von HiOPC

Um diesen Fehler zu beheben wurden die DCOM Startberechtigungen der HiOPC auf dem HiOPC Server geändert, so dass jeder Benutzer den Prozess starten darf (auch anonyme Anmeldung). Nach dieser Änderung konnten im RSView Studio Tags „gebrowst“ werden. Ein Zugriff auf die Daten war allerdings immer noch nicht möglich. Zu dem Zeitpunkt des Verbindungsversuchs trat auf dem HiOPC Server folgender Fehler auf:



Eventlog Eintrag auf dem HiOPC Server

Abb. 32: DCOM Fehler - keine Antwort vom Prozess

Um ein genaueres Bild über die Abläufe zu bekommen, wurden wieder die Prozesse auf dem HiOPC Server beobachtet. Besonderes Augenmerk galt hierbei dem Prozess des HiOPC Servers und dem Benutzer, der den Prozess gestartet hat. Es war zu erkennen, dass der Prozess mehrmals unter verschiedenen Benutzern gestartet wurde. Da dies von dem HiOPC nicht unterstützt wird, wurde die DCOM Identität für HiOPC auf einen festen Benutzer eingestellt.

7.3 Geänderte DCOM Identität

Der mehrfach gestarteten HiOPC Prozesses konnte auch mit einem lokal installierten RSView Studio nachgestellt werden. Da bei einer lokalen Installation der „interactive User“ (also der Benutzer, der zurzeit am Computer angemeldet ist) verwendet wird, ist auch die grafische Oberfläche des HiOPC aktiv. Beim Zugriff über das RSView Studio wurde der Prozess des HiOPC von FactoryTalk doppelt gestartet (einmal unter dem Konto des Computers und einmal unter dem Konto des angemeldeten Benutzers). Da die grafische Oberfläche des HiOPC aktiv war, wurde folgende Meldung angezeigt:

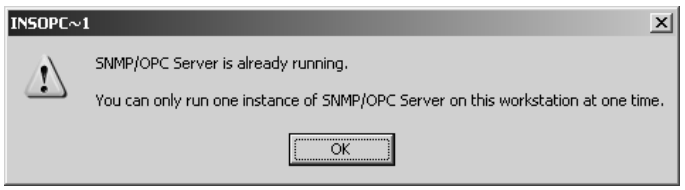


Abb. 33: Fehlermeldung des HiOPC

Diese Meldung war die Bestätigung für die Vermutung, dass von HiOPC nur eine Instanz zugelassen ist. Nach dem, schon beschriebenen, Anpassungen der Identität konnten beliebig viele Verbindungen unter verschiedensten Benutzern zu HiOPC aufgebaut werden (getestet wurden fünf Verbindungen gleichzeitig).

Um auch eventuelle Probleme bei den Zugriffberechtigungen zu verhindern, wurden diese DCOM Einstellungen ebenfalls auf Zugriff für jeden eingestellt (ähnlich den angepassten Startberechtigungen).

8 Netzwerkmanagement

Mit steigender Komplexität von Computernetzwerken wird auch der Bedarf nach Möglichkeiten der Verwaltung und Überwachung der Netzwerke größer. Zu diesem Zweck sind Netzwerkmanagementsysteme entstanden, die es erlauben, Netzwerkplanungen, Steuerungen und Fehlersuche durchzuführen. Die Aufgaben einer solchen Verwaltung sind in fünf Gebiete eingeteilt, die sich wiederum aus verschiedenen Unteraufgaben zusammensetzen:

1. **Konfiguration**
    - Parameter modifizieren
    - Aktionen starten und beenden
    - Zustand der Netzwerkkomponenten erfassen
    - Netzwerk konfigurieren
2. **Fehlermanagement**
    - Fehlermeldungen
    - Fehlerstatistik
    - Fehlerdiagnose
    - Schwellwerte für Alarmierungen
3. **Leistungsmanagement**
    - Echtzeitstatistiken
    - RMON (Remote Monitoring)
4. **Sicherheitsmanagement**
    - Passwortverwaltung
    - Zugangsverwaltung
    - Erkennen von unberechtigten Netzwerkteilnehmern
5. **Abrechungsmanagement**
    - Erfassen von Kostenanteilen (Verkehrsaufkommen pro Benutzer)
    - Umschlagen dieser Kosten

Solch ein Netzwerkmanagement setzt ein gemeinsames Protokoll voraus, über das das Endgerät und die entsprechende Netzwerkmanagementstation kommunizieren können. Das „Simple Network Managment Protokoll“ (SNMP) ist ein solches Protokoll und erlaub den herstellerübergreifenden Datenaustausch von Verwaltungsinformationen zwischen den einzelnen Geräten. Durch die

weite Verbreitung dieses Protokolls und die Verwendung durch Geräte verschiedenster Hersteller ist SNMP quasi das Standardprotokoll in diesem Bereich *[8.1]*.

Das SNMP Protokoll kann den anwendungsorientierten Schichten des OSI Referenzmodells (siehe Kapitel 2.3) zugeordnet werden.

OSI Schicht		Protokoll	
7	Anwendung	**SNMP**	Anwendungs-orientiert
6	Darstellung		
5	Sitzung		
4	Transport	UDP / TCP	Transportorientiert
3	Vermittlung	IP	
2	Sicherung	z.B. Ethernet	
1	Bitübertragung	z.B. Ethernet	

Abb. 34: Zuordnung SNMP in OSI Referenzmodell

8.1 Komponenten eines Netzwerkmanagementsystems

Die Komponenten eines Netzwerkmanagementsystems bestehen im Wesentlichen aus zwei Komponenten *[8.2]*.

1. **Agent**

   Ein Agent ist eine Einrichtung in einer Netzwerkkomponente, die Informationen für die Netzwerkverwaltung bereitstellt und auf die eigene Netzwerkkomponente einwirken kann, also auch Änderungen von Einstellungen zulässt. Unter Netzwerkkomponenten werden alle verwaltbaren Geräte wie z.B. Switche, Router oder Gateways

verstanden. Eine Netzwerkkomponente kann aber auch eine auf einem PC installierte Software sein, die über SNMP verwaltet werden kann.

2. **Manager**

   Der Manager ist ein Programm, das zur Verwaltung des Netzwerks genutzt wird. Hierzu kann auf jeden verfügbaren Agenten zugegriffen werden und die bereitgestellten Informationen können durch den Manager betrachten und bearbeitet werden. Des Weiteren kann ein Manager selber als Agent fungieren, in dem eigene oder gesammelte Informationen für andere Manager bereitgestellt werden. So kann ein vielschichtiges und zu höheren Ebenen offenes System entstehen. Im Fall der, im Rahmen dieser Arbeit verwendeten Systeme, handelt es sich bei dem Manager um die Hirschmann Software „HiVision". Ach der OPC Server der Firma Hirschmann, der den Zugriff auf SNMP fähige Geräte ermöglicht, erfüllt die Kriterien eines Managers (also lesen und schreiben von Informationen).

Zur Kommunikation wird das schon vorgestellte SNMP Protokoll verwendet. Um Informationen von einem Agenten lesen zu können, wird die „Management Information Base" (MIB) verwendet. Hierbei handelt es sich um eine Baumartige Struktur, in der Objekte, die bereitgestellt werden, eingetragen sind.

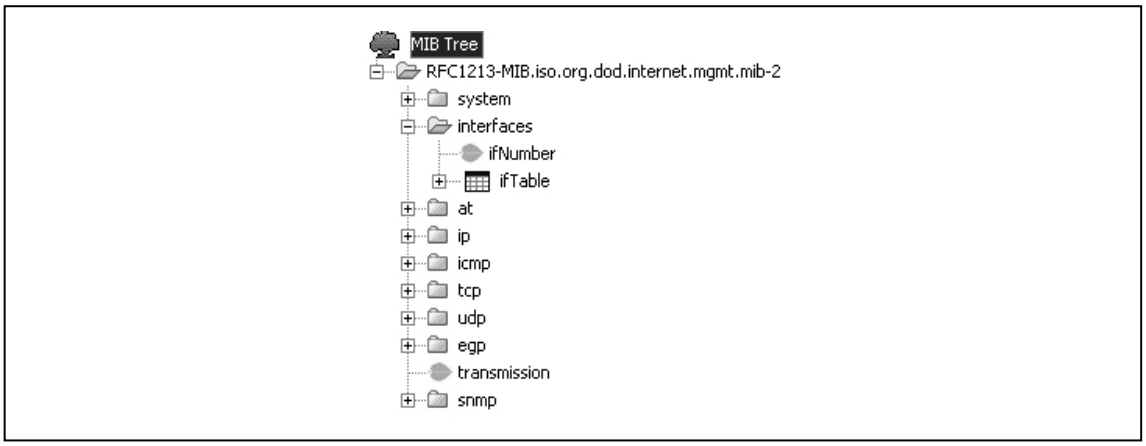


Abb. 35: MIB Baumstruktur

Die MIB enthält Informationen über die Objekte, die der Agent bereitstellt und von denen Informationen gelesen bzw. geschrieben werden können. Sofern der Manager das Gerät nicht selbständig erkennt, können verschiedene MIB

geladen werden. Es existier z.B. eine MIB, die nach dem RFC 1213 aufgebaut ist und Informationen für TCP/IP basierende Geräte enthält. Über solche Standardbibliotheken hinaus existieren auch MIB, die auf das entsprechende Gerät zugeschnitten sind. Im Falle des OPC Servers „HiOPC“ wird z.B. der Typ des Switches automatisch erkannt und eine Art automatischer Import durchgeführt um alle verfügbaren Daten anzuzeigen. Bei der Verwendung eines allgemeinen SNMP Clients (z.B. der verwendete SNMP Client iReasoning MIB Browser) muss von dem Benutzer eine bestimmte MIB ausgewählt werden.

8.2 Verwendung von Netzwerkmanagementinformationen in RSView SE

Durch den Einsatz des Produktes „HiOPC“ der Firma Hirschmann ist es möglich, Daten aus verschiedenen Netzwerkkomponenten in RSView SE nutzen zu können. HiOPC stellt die Informationen über die eingerichteten Geräte über die Standardschnittstelle OPC zur Verfügung.

Nach der Einrichtung des HiOPC Servers (siehe auch Kapitel 7) können Daten über einen OPC Client (auch RSView SE fungiert als OPC Client) gelesen und geschrieben werden. Auf dem HiOPC Server können Tags eingerichtet werden, die nach dem „polling” Verfahren aktualisiert werden. Es können Tags eingereichtet werden, die sich aus verschiedenen Werten zusammensetzen und es können Tags eingerichtet werden, die zur Alarmierung verwendet werden können (so genannte Traps bei denen das „polling“ Verfahren umgangen wird, eine Mitteilung also direkt an HiOPC gesendet wird).

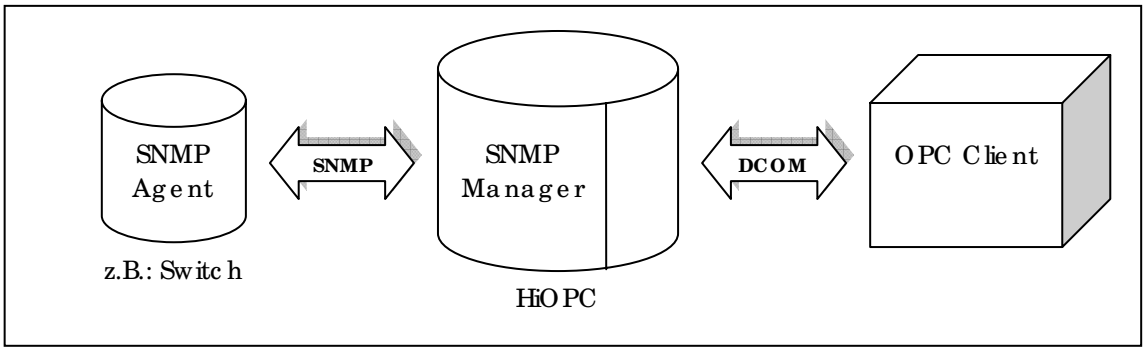


Abb. 36: Funktionsprinzip HiOPC

9 Zusammenfassung

Ein komplexes und weit ausgedehntes Visualisierungssystem erfordert umfangreiche Diagnosemöglichkeiten um Ausfällen vorzubeugen oder schnell beheben zu können. Um eine strukturierte Fehlersuche zu ermöglichen, werden von den Herstellern der einzelnen Komponenten Programme bereitgestellt, um jeweils einen Teil des Visualisierungsnetzwerks zu überwachen. Eine Übersicht über den Gesamtstatus der Visualisierungsarchitektur ist so nur schwierig möglich.

Die Visualisierungssoftware RSView SE bietet jedoch Möglichkeiten über Standardschnittstellen viele Informationen aus verschiedenen Quellen zu sammeln, auszuwerten und anzuzeigen. Aufgrund des Charakters eines verteilten Visualisierungssystems können diese Informationen von jedem Teilnehmer angezeigt werden. Diese Möglichkeiten sollen zur Diagnose der verschiedenen Komponenten genutzt werden.

Nachdem analysiert wurde, mit welchen Netzwerkteilnehmern ein RSView SE Client kommuniziert, konnten so, auf jede Komponente abgestimmte Vorgehensweisen entwickelt werden um ihren Status anzeigen zu können. Dank der Integration von VBA in RSView SE Displays dienen Visual Basic Programme als Ausgangspunkt der Verarbeitung. Durch diese weit verbreitete Programmiersprache ist es möglich auf eine umfangreiche Sammlung bestehender Bibliotheken zurückzugreifen. Außerdem lassen Informationen von dem Betriebsystem, auf dem die Anwendung läuft, auslesen und weiterverarbeiten. Die Darstellung der Ergebnisse erfolge ebenfalls über VBA in RSView SE, da auch die Displays aus dem Programm heraus animiert werden bzw. Daten in die Displays geschrieben werden können

Für die Server, die zu der RSView SE Applikation gehören, war es möglich, über das „Display Client Object Model“ Informationen über einzelne Server abzurufen. Die im Objekt Modell bestehende Funktion zu diesem Zweck wurde in eine eigene Funktion übernommen und durch die weitere Verarbeitung der ursprünglichen Daten und einer umfangreichen Fehlerbehandlung im

Funktionsumfang ergänzt. Um Informationen über die physikalischen Computer zu ermitteln, wurden zwei Konzepte entwickelt, um diese Informationen breit zu stellen, da diese Informationen innerhalb eines RSView SE Projektes nicht verfügbar sind. Nach Gegenüberstellung der zwei Konzepte wurde eine RSView SE Parameter Datei benutzt, um Information über die physikalischen Computer und den Servertyp eines Servers in der Applikation bereit zu stellen.

Ein weiterer Schritt, um den Status des Visualisierungsnetzes darzustellen, war die Einbindung des Windows Eventslogs in RSView SE. Hierzu wurde ein Konzept entwickelt, mit dem Daten aus dem Windows Eventlog in einem RSView SE Display dargestellt werden können.

Die mit den vorgestellten Techniken ausgestatteten Visualisierungsbilder befinden sich an ausgewählten Bedienstationen in der Produktion des Mercedes Sprinters (DaimlerChrysler Werk Düsseldorf, Bereich Rohbau) in Betrieb.

Anhang zur Diplomarbeit

Quellenangabe

[2.1] **Rockwell Soaftware RSView SE User's Guide**
Rockwell Automation, 2004

[2.2] **OPC Foundation**
Online unter:
http://www.opcfoundation.org/Default.aspx/01_about/01_whatis.asp?MID=AboutOPC
Entnommen am: 02. März 2007, 17:00 Uhr

[2.3] **Iwanitz, Frank / Lange, Jürgen:**
OPC: Grundlagen, Impementierung und Anwendung, 2. Auflage
Heidelberg: Hüthig Verlag, 2002

[2.4] **Grundlagen Local Area Network (LAN)**
Handbuch
Hirschmann Electronics GmbH & Co. KG

[2.5] **RRZN / Universität Hannover**
Netzwerke – Grundlagen, 3. Auflage
Hannover: RRZN, 2000

[2.6] **Microsoft Technology Overview VBA**
Online unter:
http://www.microsoft.com/europe/vba/prodinfo/german/
Entnommen am: 02. März 2007, 17:00 Uhr

[2.7] **MSDN Library „Set Timer" Funktion**
Online unter:
http://msdn2.microsoft.com/en-us/library/ms644906.aspx
Entnommen am: 02. März 2007, 17:00 Uhr

[2.8] **MSDN Library „Kill Timer" Funktion**
Online unter:
http://msdn2.microsoft.com/en-us/library/ms644903.aspx
Entnommen am: 02. März 2007, 17:00 Uhr

[2.9] **Bünning, Uwe / Krause, Jörg:**
Windows Server 20003,
München: Markt+Technik Verlag, 2003

[2.10] **Windows Management Instrumentation**
Online unter:
http://msdn2.microsoft.com/en-us/library/aa394582.aspx
Entnommen am: 02. März 2007, 17:00 Uhr

[5.1] **MSDN Library „IsDestinationReachable" Funktion**
Online unter:
http://msdn2.microsoft.com/en-us/library/aa376851.aspx
Entnommen am: 02. März 2007, 17:00 Uhr

[5.2] **Martin, René:**
XML und VBA lernen
Bonn: Addison Wesley Verlag, 2002

[6.1] **WQL (SQL for WMI)**
Online unter:
http://msdn2.microsoft.com/en-us/library/aa394606.aspx
Entnommen am: 02. März 2007, 17:00 Uhr

[8.1] **Internet Engineering Task Force - SNMP**
Online unter:
http://www.ietf.org/rfc/rfc1157.txt
Entnommen am: 02. März 2007, 17:00 Uhr

[8.2] **Grundlagen Local Area Network (LAN)** *(wie [2.4])*
Handbuch
Hirschmann Electronics GmbH & Co. KG

Abbildungsverzeichnisse

Quellcode

Abbildungen

Erklärung zur Diplomarbeit

Name:	Anjarwalla
Vorname:	Armin
Matr.-Nr.:	1155236
Studiengang:	Angewandte Automatisierungstechnik

An den Prüfungsausschuss
des Fachbereichs Automatisierungstechnik
der Universität Lüneburg
Volgershall 1

21339 Lüneburg

Erklärung zur Diplomarbeit

Ich versichere, dass ich diese Diplomarbeit – bzw. meinen Teil, der als Gruppenarbeit vergebenen Diplomarbeit – selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Lüneburg, den 06.03.2007

Armin Anjarwalla