Bachelorarbeit

Christian Pflüger

Aufbau einer Infrastruktur zur Überwachung und Analyse von Clients mittels .NET Webservices und Windows Vista Gadgets beim TÜV NORD

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Christian Pflüger

Aufbau einer Infrastruktur zur Überwachung und Analyse von Clients mittels .NET Webservices und Windows Vista Gadgets beim TÜV NORD

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuende Prüferin : Prof. Dr. rer. nat. Bettina Buth
Zweitgutachter : Prof. Dr.-Ing. Martin Hübner

Abgegeben am 13. März 2008

Christian Pflüger

Thema der Bachelorarbeit

Aufbau einer Infrastruktur zur Überwachung und Analyse von Clients mittels .NET Webservices und Windows Vista Gadgets beim TÜV NORD

Stichworte

Web Service, Windows Vista, Gadget, .NET, COM, UDDI, pro-aktiv

Kurzzusammenfassung

Für ein Unternehmen ist es wichtig, dass seine Mitarbeiter unterbrechungsfrei arbeiten können. Um den Ausfall von Arbeitszeit durch Computerprobleme so gering wie möglich zu halten, ist es erforderlich, Probleme pro-aktiv erkennen zu können, entsprechende Gegenmaßnahmen einzuleiten und damit die möglichen Probleme verhindern zu können.

Diese Arbeit beschreibt die Entwicklung eines Systems zur pro-aktiven Erkennung von möglichen Problemen auf Client Computern. Um dies zu gewährleisten, werden für diesen Zweck geeignete Daten auf dem Client mit Referenzdaten einer Datenbank verglichen. Die Ergebnisse werden den Nutzern durch ein Windows Vista Gadget sowie den Administratoren über einen repräsentativen Gesamtwert mitgeteilt, damit, wenn nötig, entsprechende Gegenmaßnahmen eingeleitet werden können.

Christian Pflüger

Title of the paper

Development of an infrastructure for monitoring and analysis of clients using .NET Web Services and Windows Vista Gadgets

Keywords

Web Service, Windows Vista, Gadget, .NET, COM, UDDI, pro-active

Abstract

For a company it is important that all employees can do their work without any interrupts. To reduce the risk of loss of working time due to computer problems, it is necessary to identify these problems before their appearance to initiate appropriate counteractive measures and thus avoid possible problems.

This thesis describes the development of a system allowing to identify possible problems on a client computers before their occurrence. To achieve this ability, appropriate values from the client are compared with reference data from a database. The results are shown to the users on a Windows Vista Gadget and to the administrators by a value that indicates the necessity to initiate appropriate counteractive measures.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Zielsetzung

Das Ziel dieser Arbeit ist die Erstellung einer Infrastruktur zur Überwachung und Analyse von Clients mittels .NET Webservices und eines Windows Vista Gadgets. Die Arbeit soll dazu beitragen, Probleme auf Clients pro-aktiv zu verhindern. Dabei werden die Anwender der Clients durch die Nutzung des Gadgets aktiv mit einbezogen. Es wird eine Infrastruktur entwickelt, durch die eine Analyse sowie eine Überwachung der Clients ermöglicht wird. Die Analyse beruht auf einem Vergleich: Es werden verschiedene Parameter der Clients mit entsprechenden Referenzwerten verglichen. Die Referenzwerte für den Vergleich werden über einen Web Service bereitgestellt. Durch den Vergleich wird auf dem Client in der Registry ein Wert erzeugt, der über die laufende Inventarisierung importiert werden kann. Hiermit wird die Überwachung möglich. Ist durch diesen Vergleich ersichtlich, dass sich in naher Zukunft Probleme mit/auf dem entsprechenden Client ergeben, so wird dies dem Nutzer durch das Gadget mitgeteilt. An der Ausprägung des importierten Wertes können Administratoren feststellen, ob auf dem Client demnächst ein Problem zu erwarten ist. Hierauf kann dann entsprechend reagiert werden.

1.2 Gliederung der Arbeit

Diese Arbeit umfasst fünf Kapitel. Im ersten Kapitel wird eine kurze Vorstellung des TÜV NORD gegeben sowie die Motivation für diese Arbeit dargestellt. Das zweite Kapitel beschreibt die in dieser Arbeit zum Einsatz gekommenden Technologien. Im dritten Kapitel werden die Anforderungen an dieses Projekt beschrieben sowie verschiedene Möglichkeiten der Realisierung dargestellt. Das vierte Kapitel beschreibt die im Laufe dieser Arbeit entstandenen Realisierungen. Im fünften Kapitel wird schließlich ein Ausblick über die weiteren Möglichkeiten dieser Arbeit gegeben und ein Fazit zu den Problemen während dieser Arbeit gezogen.

1.3 Motivation

1.3.1 Beschreibung der Ausgangssituation

Die Ausgangssituation stellt sich wie folgt dar: Unternehmensweit werden Betriebssysteme von Microsoft eingesetzt, auf der Client- sowie der Serverseite. Aktuell wird die Einführung bzw. Umstellung auf Windows Vista und Windows Server 2008 vollzogen. Die Softwareverteilung basiert auf Microsoft's SCOM (System Center Operations Manager). Ebenso wird hierdurch die Hardware- sowie Softwareinventur durchgeführt, wie auch die Überwachung der Serversysteme. Microsoft's SQLServer stellen die Datenbanksysteme. Das ERP (Enterprise Resource Planning) -System wird durch SAP-ERP realisiert. HPOpenView Servicedesk wird als Callsystem mit Auftragsverwaltung bis hin zur Lösungsdatenbank eingesetzt.

Der typische Nutzer eines Clients in der TÜV NORD Gruppe (im folgenden TÜV-Nutzer genannt) kann wie folgt charakterisiert werden. Er benötigt seinen PC als Grundlage für effektives Arbeiten, zum Teil auch für seine Prüftätigkeiten. Aus diesem Grunde muss der Computer funktionieren und sollte möglichst keine Probleme bereiten, die zu Ausfallzeiten führen. Sollte es doch einmal zu Problemsituationen kommen, müssen diese schnellstmöglich behoben werden, damit es nicht zum Ausfall von Arbeitszeit kommt. Der TÜV-Nutzer ist also bemüht um ein uneingeschränkt lauffähiges System, damit er erfolgreich seine Arbeit verrichten kann. Diesen Zustand möchte er mit möglichst wenig Aufwand erreichen bzw. halten.

Sollte beim TÜV-Nutzer ein Problem auftreten, das er selbst nicht zu lösen imstande ist, hat er die Möglichkeit sich telefonisch sowie per Mail an die ServiceLine zu wenden. Diese nimmt die Anfrage des TÜV-Nutzers auf und hilft ihm sofort weiter.

Für den TÜV-Nutzer gibt es eine Anlaufstelle im Intranet, um Informationen über sein Gerät einzuholen: das Ampelsystem. Um Auskünfte von diesem System zu erhalten, muss der TÜV-Nutzer es gezielt im Intranet aufrufen.

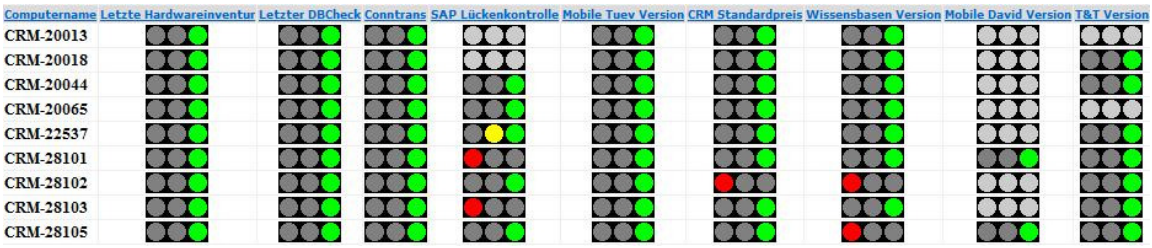

Abbildung 1.3.1-1: Ampelsystem mehrerer Clients
grün: gut; gelb-grün: brauchbar; gelb: zulässig; rot: unzulässig; grau: nicht verfügbar

In dieser Abbildung sind zur Verdeutlichung mehrere Clients abgebildet, ein TÜV-Nutzer bekommt immer nur seinen eigenen angezeigt. Diese Übersicht stellt einen Ausschnitt aus einer Regionalansicht (alle Clients einer Region) dar. Diese Ansicht steht nur den Regionalleitern zur Verfügung.

Für das Ampelsystem sind folgende Daten relevant:

- Wann fand die letzte Inventur statt?
- Wann wurde der letzte Datenbank-Selbsttest durchgeführt und wie ist das Ergebnis?
- Wurden alle Rechnungen übermittelt?
- Wie aktuell sind die Datenbanken?
- Wie ist der Stand der Preisliste?
- Welche Versionen einzelner Software Pakete sind installiert?

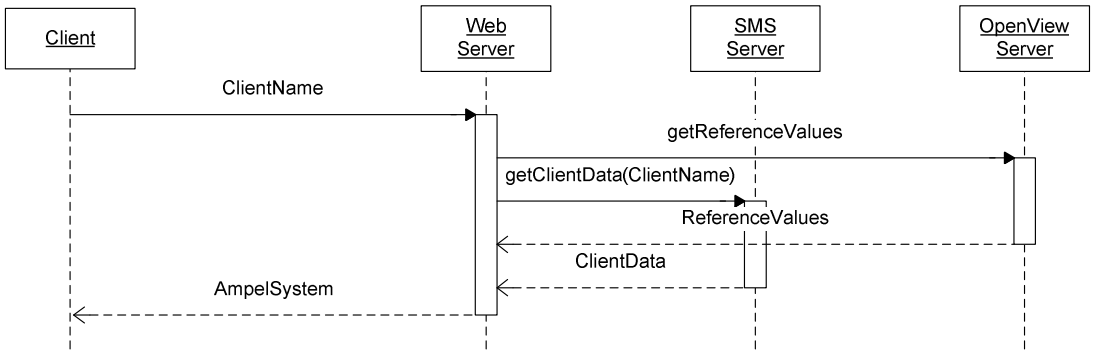

Abbildung 1.3.1-2: Abruf clientspezifisches Ampelsystem

Sobald ein Client die Seite mit dem Ampelsystem aufruft, schickt er seinen Namen an den Server. Dieser vergleicht nun die Clientdaten aus einer SCOM-Datenbank mit aktuellen Referenzwerten und stellt das Ergebnis in Form einer Ampel dar. Die

Referenzwerte stammen von einem virtuellen Computer, der nur zu diesem Zweck unter HP OpenView angelegt worden ist. Die Clientdaten basieren auf der SCOM Hard-/sowie Software Inventur und den aktuellen Daten aus den SAP Systemen. Durch die Inventur werden diversen Hardware- sowie Software Informationen gesammelt, unter anderem auch die relevanten Informationen für die Ampelanzeige. Die Inventarisierung auf dem Client wird in drei Situationen angestoßen: bei Ausführung des Logon-Skriptes (d.h. wenn sich ein TÜV-Nutzer am Client anmeldet), wenn kein Neustart des Clients erfolgt (und damit keine Anmeldung eines TÜV-Nutzers) alle 24 Stunden sowie nach der Installation von Softwarepaketen auf dem Client.
Eine statistische Auswertung der Daten aus dem Ampelsystem kann von berechtigten Personen abgerufen werden. Diese Statistik gibt den aktuell in SCOM enthaltenen Stand wieder und bezieht sich auf die Gesamtheit aller Clients. Ebenso ist es für entsprechende Personen möglich, sich alle Clients ihres Bereiches/ihrer Region detailliert anzeigen zu lassen (siehe Abbildung1.3.1-1). Auch diese Übersicht gibt den aktuellen SCOM Stand wieder.
Auf der Basis von Microsoft‘s SCOM ist ein pro-aktives Verfahren zur frühzeitigen Erkennung von Schreib-/Lesefehlern auf Notebook Festplatten entwickelt worden und bereits seit über einem Jahr im Einsatz. Es wird ein SCOM-Client auf den Geräten installiert. Dieser meldet ab einem festgelegten Schwellwert an Schreib-/Lesefehlern an einen Server, das die Festplatte getauscht werden muss. Der Server wiederum informiert per ServiceCall (Ticket im Helpdesk System) die Service Line, dass die Festplatte defekt ist. Daraufhin ruft ein Mitarbeiter der ServiceLine beim entsprechenden Kunden an und bittet ihn, sofern möglich, seine Daten zu sichern, da er in den nächsten zwei Werktagen ein Ersatzgerät zugeschickt bekommt. [MicrKun1]

1.3.2 Defizite der Ausgangssituation

Aufgrund der Tatsache, dass die Aktualisierung der SCOM-Datenbank mit Inventarisierungsdaten schlimmstenfalls alle 24 Stunden angestoßen wird, können die zugrunde liegenden Daten für das Ampelsystem in den meisten Fällen nicht hoch aktuell sein. Dies bedeutet, dass das Ampelsystem keine hochgradig aktuelle Übersicht bieten kann, sondern immer nur basierend auf den letzten Inventarisierungsdaten. Es ist

nicht möglich, den Jetzt-Zustand eines Clients ohne weitere Maßnahmen (Logon, Software-Pakete installieren) abzurufen.

Es findet keine Gewichtung der angezeigten Ergebnisse statt. Muss der TÜV-Nutzer erst bei einer roten Ampel tätig werden oder ist es auch schon bei einer gelben angebracht. Oder kommt es darauf an, in welcher Kategorie sich die Ampel befindet?

Das Ampelsystem ist nicht pro-aktiv. Der TÜV-Nutzer wird nicht drüber informiert, wann er tätig werden müsste, um sein Gerät weiterhin problemfrei betreiben zu können. Er muss selbst tätig werden, um Informationen über sein Gerät abzurufen. Wenn er dies tut, kann es schon zu spät sein. Es ist eventuell bereits ein Problem aufgetreten, das er bisher nicht bemerkt hat. Nun muss im Nachhinein das Problem mit seinen Auswirkungen gelöst werden.

Das Ampelsystem ist nicht in der Lage, verschiedene Kategorien von Geräten zu unterscheiden. Alle Geräte werden bezüglich derselben Merkmale analysiert. Dies bedeutet, dass es nur für einen beschränkten TÜV-Nutzerkreis sinnvoll ist, das Ampelsystem für ihre Geräte zu verwenden. Wenn relevante Merkmale einer Gerätekategorie nicht berücksichtigt werden, dafür allerdings Merkmale untersucht werden, die für das entsprechende Gerät gar nicht in Betracht kommen, wird das System hier zu Recht keine Anwendung finden.

Um auf das Ampelsystem zugreifen zu können, muss der TÜV-Nutzer online und im TÜV Netzwerk eingewählt sein. Ansonsten steht ihm das Ampelsystem nicht zur Verfügung. Ähnlich verhält es sich mit der frühzeitigen Erkennung von Schreib-/Lesefehlern auf den Notebook Festplatten: der SCOM Client kann seine Daten nur an den Server übermitteln, wenn das Gerät über einen Internetzugang verfügt und eingewählt ist.

Um das Ampelsystem zu nutzen, muss der TÜV-Nutzer es gezielt im Intranet aufrufen. Er muss also manuell eine Adresse eingeben, einen Favoriten aufrufen oder sich durch das Intranet klicken. Dies führt dazu, dass nur wenige TÜV-Nutzer regelmäßig in entsprechend kurzen Intervallen das System aufrufen und darüber den Status ihres Clients abfragen.

1.4 Firmenportrait - Die TÜV NORD Gruppe

Abbildung 1.4-1: TÜV NORD Logo

1.4.1 Wie alles begann

Im Zuge der fortschreitenden Industrialisierung des 19.Jahrhunderts wurde im Jahre 1868 in Hamburg der Norddeutsche Verein zur Überwachung von Dampfkesseln (DÜV) gegründet. In wechselvoller Geschichte änderte sich der Name in TÜV Hamburg und zu TÜV Norddeutschland. In der Folge der Wiedervereinigung wurde im Jahr 1993 die Zusammenführung mit dem TÜV Nord in Rostock erreicht, dessen Name für das gemeinsame Haus übernommen wurde. Im Jahre 1997 wurde eine Vereinbarung über den Zusammenschluss mit dem TÜV Hannover / Sachsen-Anhalt zur TÜV NORD GRUPPE unterzeichnet. Die Fusion mit dem Rheinisch-Westfälischen TÜV (RWTÜV) wurde im Jahr 2004 realisiert.

1.4.2 Firmenprofil

Die TÜV NORD Gruppe [TÜVNORDW] ist einer der größten technischen Dienstleister in Deutschland. Hauptstandorte sind Hamburg, Hannover und Essen.

8000 Mitarbeiter setzen ihren Sachverstand unabhängig und neutral in allen Fragen der technischen Sicherheit, des Umweltschutzes und der Zertifizierung von Systemen und Produkten ein. Insbesondere in den Bereichen Anlagentechnik, Energie- und Systemtechnik, Fahrzeugprüfungen und Fahrerlaubnis, Aus- und Weiterbildung. Der TÜV NORD ist weltweit in 70 Ländern aktiv.

Die folgenden drei Abbildungen zeigen, in Form von Organigrammen, die Struktur der TÜV NORD Gruppe – der Konzern, die Service GmbH, der EDV-Bereich. Die vorliegende Bachelorarbeit wurde in der Abteilung Client Management unter der Projektleitung von Herrn Frank Boerger erstellt.

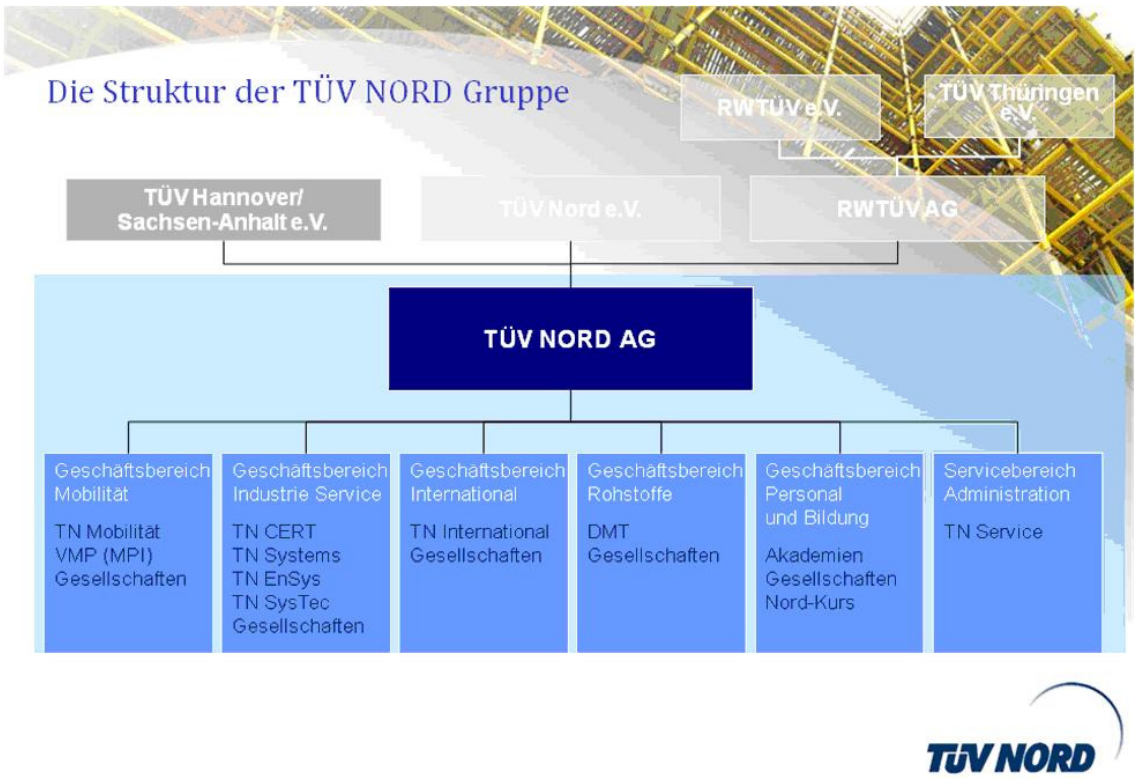

Abbildung 1.4.2-1: Die TÜV NORD Gruppe – der Konzern

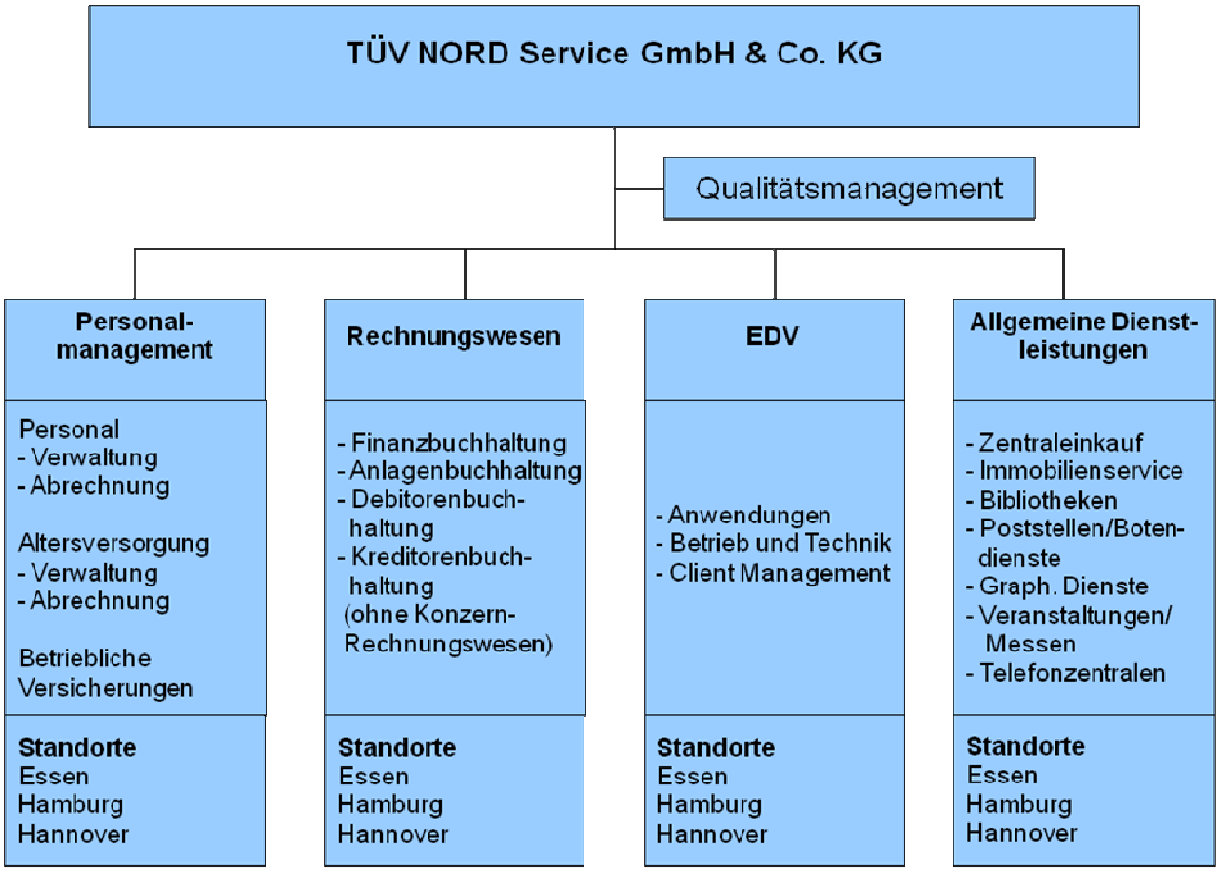

Abbildung 1.4.2-2: Die TÜV NORD Gruppe - die Service GmbH

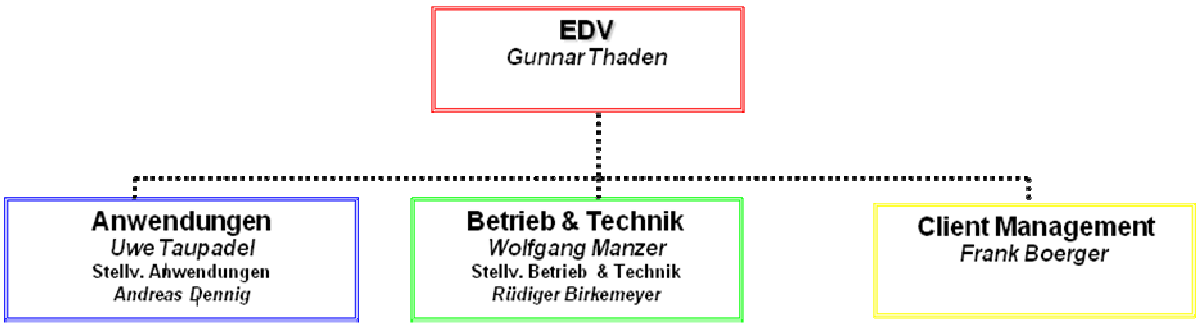

Abbildung 1.4.2-3: Die TÜV NORD Gruppe - der EDV-Bereich

1.4.3 Der IT-Bereich des TÜV NORD

Der IT-Bereich des TÜV NORD betreibt und betreut eine Reihe von Systemen mit zugehörigen Anwendungen. Sie stehen an unterschiedlichen Standorten und unterscheiden sich nach:

- Systemen zum Test,
  die für Entwicklungsarbeiten im IT-Bereich zur Verfügung stehen,
- Systemen zur Qualitätssicherung,
  die für Integrationstests der internen Kunden genutzt werden,
- Systemen im produktiven Einsatz,
  die im Notfall auf die Qualtätssicherungssysteme ´umgesetzt´ werden können.

Die Systemlandschaft ist aufgebaut aus:

- Basis-Infrastruktur (LAN-, WAN-, Internet-Leitungen, Switche, Router, Server) zur Realisierung der internen und externen Kommunikation, inkl. Firewall-, SingleSignOn-, Datensicherungs- und Archivierungs-Aktivitäten.
- Anwendungs-Infrastruktur (Desktops, Notebooks, Tablet-PCs, Betriebssoftware) inkl. der Methoden zur Softwareverteilung und Betriebsüberwachung.
- Anwendungssysteme (Datenbanken, Anwendungssoftware) zur Gestaltung der Kundengeschäftsprozesse, inkl. Berechtigungsprüfungen sowie aller Sicherheits- und Datenschutz-Erfordernisse.

Der IT-Bereich in der TÜV NORD Gruppe hat die strategische Entscheidung getroffen, mit wenigen starken Partnern zusammen zu arbeiten, um in allen Bereichen die Spitze der technologischen Entwicklung zu halten:

- Hardware HP, Lenovo, Cisco
- Systeme Microsoft – Longhorn, Vista, SQL-Server, SharePoint
- Anwendungen SAP – MySAP ERP 2005, BW, DUET, Portale

Bei Microsoft nimmt der TÜV an verschiedenen TAP (Technical Adoption Partner)-Programmen teil. [MicrTAP] [mbufMicr]

Bei SAP ist der TÜV in einer Reihe von RAMP-UP-Programmen (Markteinführungsprozess für SAP-Lösungen) vertreten und außerdem als CCC (Customer Competence Center) zertifiziert. [SAPRampU]

Einige Kennzahlen verdeutlichen die Größenordnung:

- 400 Server im Netz
- 7000 PCs im Netz (1500 Neugeräte pro Jahr)
- 1200 Druckertreiber im Einsatz
- 750 Anwendungen in der Softwareverteilung
- 100 TByte SAP-Daten

Aufgabe der IT ist es, einen störungsfreien Betrieb und die kontinuierliche Entwicklung zu gewährleisten. Dafür können Arbeitsabläufe (Prozesse) eingerichtet werden, die sich an den ´Best Practice´- Vorschlägen der ITIL (Information Technology Infrastructure Library) [itSMFITL] orientieren. Zentrale Gedanken dieser Vorschläge sind:

- die IT-Services an den gegenwärtigen und zukünftigen Anforderungen des Unternehmens und seiner Kunden auszurichten,
- die Qualität der IT-Services zu verbessern,
- Arbeitsabläufe und Informationsaustausch transparent zu gestalten,
- die Kosten der Service-Tätigkeit zu reduzieren.

ITIL wurde im Auftrag des britischen Verteidigungsministeriums entwickelt. Seit 2005 gibt es die internationale Norm20000 [ISO20000]. Auf dieser Basis kann ein Unternehmen die Arbeitsprozesse seines IT-Betriebes auch zertifizieren lassen. [WikipITI]

Im IT-Bereich des TÜV NORD
ist eine Reihe von Arbeitsprozessen bereits entsprechend ITIL eingerichtet:

- Service Desk
  Einheitliche Kontaktadresse für die Benutzer und Schnittstelle zur IT.
- Change Management und Release Management
  Einsatz standardisierter Methoden zur effizienten Produktivsetzung von Veränderungen an Systemen und Anwendungen.

- Incident Management
  Widerherstellung des normalen Servicebetriebes nach minimalen Störungen.
- Problem Management
  Minimierung der Folgen von Störungen und proaktives Vermeiden solcher Probleme.

2 Verwendete Technologien

2.1 Web Service

Definition:
*A **Web service** is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.* [W3CWebS1]

Durch Web Services wird es möglich, dass elektronische Dienste auf einfache Weise (öffentlich) zugänglich gemacht werden. Funktionalität wird genauso einfach und universell zur Verfügung gestellt, wie es heute schon mit Informationen durch Webseiten geschieht. So erstaunt es auch nicht, dass Web Services auf HTTP-Basis ähnlich wie Web Seiten funktionieren. Der Aufruf eines Web Services entspricht der Client/Server-Aufrufsequenz zwischen einem Browser und einem Web Server. Eine Anfrage wird vom Client als HTTP-POST Request an den Web Server gesendet. In dieser Anfrage ist eine SOAP-Nachricht (SOAP: wird seit Version 1.2 nicht mehr als Akronym gebraucht → siehe 2.1.1) enthalten, in der wiederum die URL (Uniform Resource Locator) und der Aufruf des Web Services sowie die Parameter für den Aufruf enthalten sind. Der Web Server extrahiert aus der Anfrage die SOAP-Nachricht, konvertiert sie in ein Format, das vom aufgerufenen Web Service verarbeitet werden kann und leitet sie an den Web Service weiter. Der Service verarbeitet die Daten und gibt ein eventuelles Ergebnis an den Server zurück. Dieser verpackt die Antwort wiederum in eine SOAP-Nachricht und sendet sie als HTTP-Response zurück an den Aufrufer. Der Client packt die SOAP-Nachricht aus, um eventuelle Daten des Ergebnisses weiterzuverarbeiten. (siehe Abbildung 2.1-1)

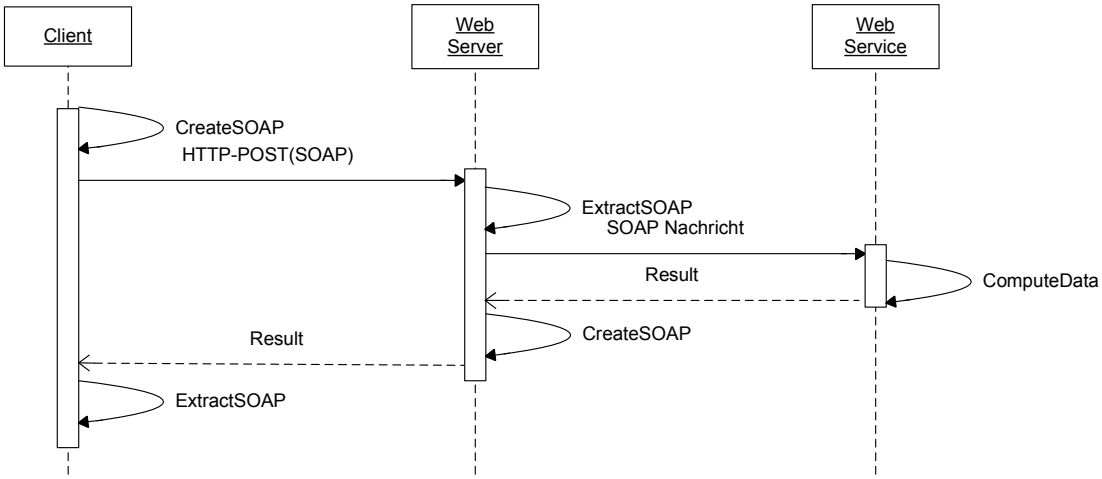

Abbildung 2.1-1: Aufruf eines WebServices

Das W3C (World Wide Web Consortium) hat eine Arbeitsgruppe eingerichtet, die sich mit den Fragen der Web Services Architektur und den entsprechenden Szenarien beschäftigt. [W3CWebS2]

Web Services verwenden XML (eXtensible Markup Language)-basierte Standards, um miteinander zu kommunizieren beziehungsweise Daten miteinander auszutauschen. So können Web Service-Nachrichten auch ohne menschliches Zutun ausgetauscht werden. Die Web Service-Architektur ist nicht an ein bestimmtes Protokoll gebunden, heute üblich sind aber TCP/IP und HTTP. Abbildung 2.1-2 gibt einen nicht vollständigen Überblick über die Aufgaben und entsprechenden Web-Services-Protokolle. [Lieb2007] [Dust2003]

Beschreiben	WSDL [W3CWebS4]
Veröffentlichen	UDDI [OASISUDD]
Suchen/Finden	UDDI [OASISUDD]
Binden	SOAP [W3CSOAP]
Aufrufen	SOAP [W3CSOAP]
Sicherheit	WS Security [OASISWe1]
Routing	WS Addressing [W3CWebS3]

Abbildung 2.1-2: Web-Services - Aufgabe und Protokolle

Die Web Service-Architektur basiert auf drei grundlegenden Bestandteilen. Zum einen auf der Client(Service Requestor)/Server(Service Provider)-Technologie, zum anderen auf einem Verzeichnisdienst(Service Broker). Er ist vergleichbar mit der Rolle eines DNS Servers (Directory Name Service), denn er vermittelt zwischen dem Client und dem Server. [Lieb2007]

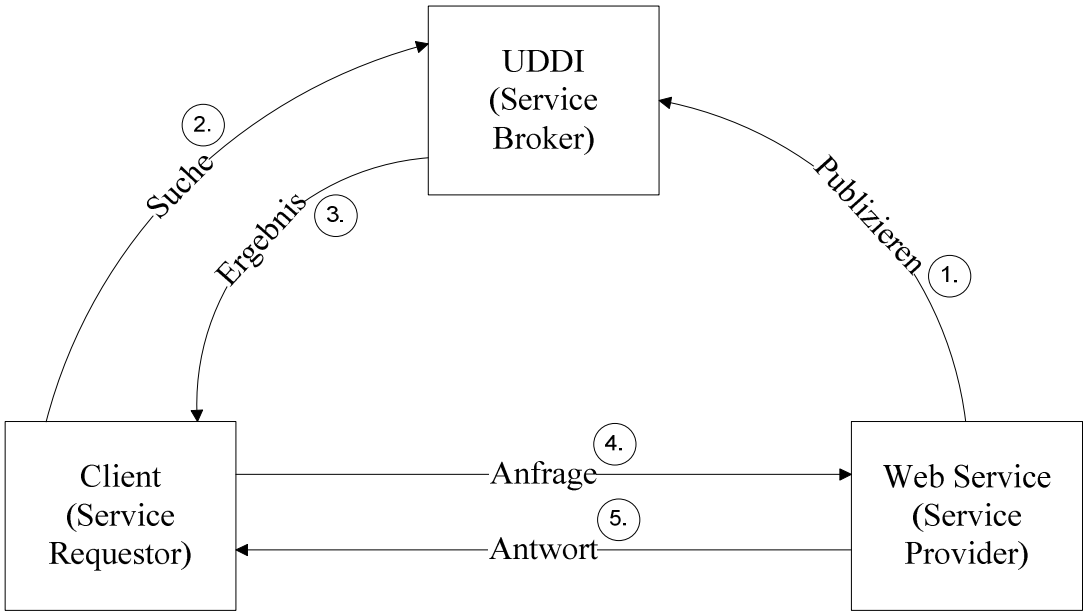

Abbildung 2.1-3: Rollenverteilung in einer Web Service Architektur

2.1.1 SOAP

SOAP (wird seit Version 1.2 nicht mehr als Akronym gebraucht) ist ein Protokoll für den Austausch von XML basierten Daten über ein Netzwerk. Dabei ist es nicht auf ein bestimmtes Transportprotokoll angewiesen. Häufig wird allerdings HTTP verwendet. Ein Vorteil von HTTP ist, dass hierdurch eine Kommunikation über Firewalls hinweg ermöglicht wird. Meist ist der Port 80 in Firewalls freigeschaltet, da über diesen Port standartmäßig die Kommunikation mittels HTTP, also zum Beispiel mit dem Internet, erfolgt. Also müssen an der Firewall keine Änderungen für Programme vorgenommen werden, die mittels HTTP kommunizieren. SOAP ist standardisiert und wird vom W3C verwaltet. Aktuell ist Version 1.2.

SOAP definiert einen Umschlag (Envelope), in dem diese XML-basierten Daten verpackt werden. Die Struktur einer SOAP-Nachricht wird in Abbildung 2.1.1-1

verdeutlicht. Eine Nachricht, die im SOAP-Format ausgetauscht wird, besteht aus zwei Teilen:

- Der SOAP-Header ist optional. Er kann Meta-Informationen über Sicherheit, Routing sowie Zusatzinformationen für eine korrekte Verarbeitung des SOAP-Bodys beinhalten. Er ist also anwendungsabhängig. Die Informationen sind auf Elementblöcke aufgeteilt. Sie werden durch eine eindeutige URI (Uniform Ressource Identification) identifiziert. Hierdurch können sie einzelnen Zwischenknoten (Intermediary) zugeordnet werden. Diese Knoten befinden sich auf der Vermittlungsstrecke zwischen Sender und Empfänger. Die Meta-Informationen können von den Zwischenknoten gelesen, hinzugefügt, gelöscht oder einfach weitergeleitet werden.
- Der SOAP-Body enthält die zu transportierenden XML-Nutzdaten für den endgültigen Empfänger.

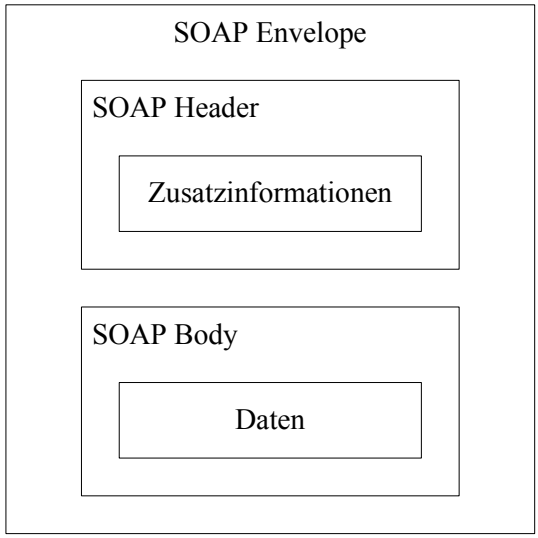

Abbildung 2.1.1-1: Aufbau von SOAP Nachrichten

Der Nachteil von SOAP in Verbindung mit XML ist die vergleichsweise datenintensive, ineffiziente Kodierung der Parameter, ähnlich der Übertragung von HTML Seiten. Dies gilt insbesondere für Binärdaten, die in eine Base64-Kodierung übertragen werden müssen [RFC 4648]. Außerdem erfordert die Kodierung sowie Dekodierung, aufgrund der XML Basis, im Vergleich mit anderen RPC (Remote Procedure Call) Protokollen mehr Rechenzeit. Ebenso werden eine automatische Speicherverwaltung und die

Verwendung von Referenzparametern wegen der üblichen Heterogenität der beteiligten Systeme nicht unterstützt. [Lieb2007] [Schi2007] [Hamm2005]

2.1.2 WSDL

Die WSDL (Web Service Description Language) beschreibt Schnittstellen von Web Services. WSDL Dokumente setzten auf SOAP auf und sind auch in XML spezifiziert. WSDL Dokumente beinhalten sowohl eine Spezifikation von Methodenschnittstellen als auch sonstige technische Daten des Web Services. Die technischen Daten dienen sowohl zum Auffinden des Services als auch zur konkreten Abwicklung der Kommunikation bei Methodenaufrufen. WSDL stellt den genauen Namen, die Anzahl und die Typen der Parameter transparent zur Verfügung, so dass eine durch sie beschriebene Methode lokal sowie entfernt aufgerufen werden kann. Informationen, die durch ein WSDL Dokument übermittelt werden, sollten ebenso in einer Software Dokumentation enthalten sein (z.B. Name, Anzahl und Typen der Parameter einer Methode(s.o.)), werden in diesem Fall allerdings auch für Computer verständlich formuliert, wodurch diese in der Lage sind, die Methoden auszuführen.
WSDL ist standardisiert und wird vom W3C verwaltet. Aktuell ist Version 2.0.
[Lieb2007] [Schi2007][Dust2003]

2.1.3 UDDI

UDDI (Universal Description Discovery and Integration) ist ein Verzeichnisdienst für Web Services, der über eine Web Service Schnittstelle sowie eine URI angesprochen werden kann. Ziel von UDDI ist ein Verzeichnisdienst, in dem man Dienste registrieren und Anwendungen flexibel Dienste suchen können. Hiermit wird eine Voraussetzung geschaffen, um Dienste dynamisch zu suchen und zu binden.
Das UDDI-Verzeichnis bildet zwar einen Standard der Web Services, gehört jedoch nicht zu den durch das W3C festgelegten Standards. Es wird durch OASIS (Organization for the Advancement of Structured Information Standards) spezifiziert und verwaltet. UDDI baut auch auf SOAP auf und wird damit durch XML beschrieben.
Informationen werden von UDDI ähnlich den Typen von Telefonbüchern gegliedert: White, Yellow und Green Pages.

- White Pages enthalten Informationen über den Anbieter von Diensten, etwa Firmenname und Kontaktinformationen.
- Yellow Pages ermöglichen eine Suche nach Branchen.
- Green Pages enthalten konkrete Dienstbeschreibungen, zum Beispiel die URI, an der der Service verfügbar ist, sowie das WSDL Dokument des Services und eine textuelle Beschreibung der Dienstsemantik.

Aufgrund dieser Informationen ist es Nutzern von Web Diensten möglich, den für sie passenden Dienst mit spezifischen Detailangaben zu suchen, zu finden und eine Verbindung mit ihm herzustellen. [Schi2007] [Eber2003]

In der folgenden Abbildung wird eine knappe, nicht vollständige Übersicht über die UDDI-Bestandteile gegeben und welche Informationen diese beinhalten.

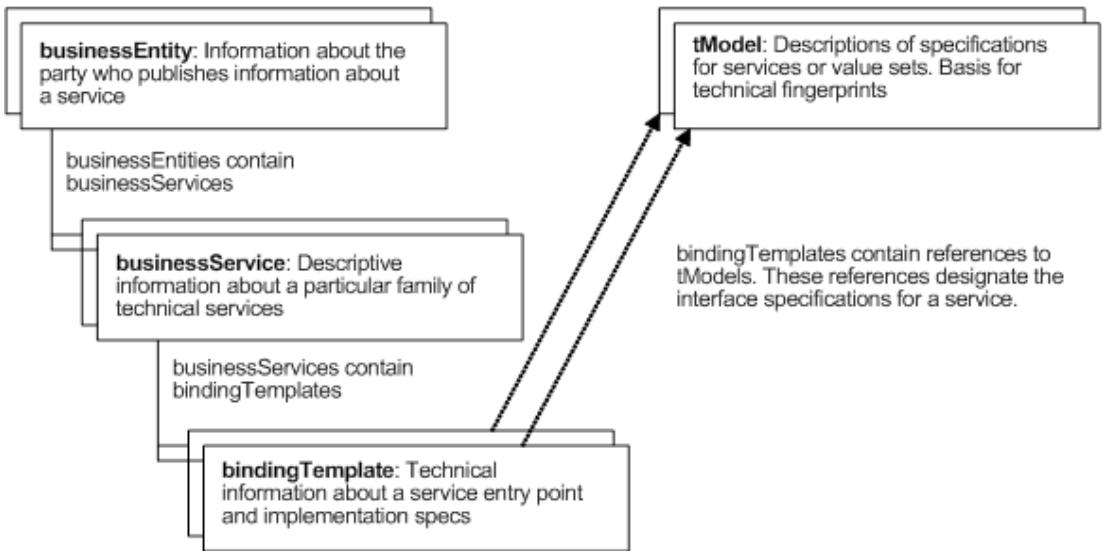

Abbildung 2.1.3-1: UDDI Bestandteile [OASISUDI]

Ein Eintrag in einem UDDI Verzeichnis besteht aus:

- businessEntity: Organisation, die den Webservice anbietet (inklusive Kontaktinformationen)
- businessService: Webservice-Gruppe, Identifikation und Klassifikation des Webservices (durch tModel's)
- bindingTemplate: technische Informationen eines Webservices(z.B. URI des Webservices), Referenzen auf tModels.
- tModel (technical model): technische Spezifikation eines Webservices, Schnittstellenbeschreibung, Klassifikationsschemata

Mit den Bestandteilen eines UDDI Eintrages ist es möglich, ein WSDL Dokument zu generieren, mit dem auf entsprechende Webservices zugegriffen werden kann. [OASISUDI]

2.2 Microsoft

2.2.1 .NET

.Net besteht hauptsächlich aus zwei Komponenten: einer Laufzeitumgebung (virtuelle Maschine) und einer objektorientierten Klassenbibliothek (Base Class Library - BCL). Programme können für .NET in diversen, entsprechend angepassten, Sprachen, programmiert werden (zum Beispiel C#, VB.NET, J#, VC++, …).

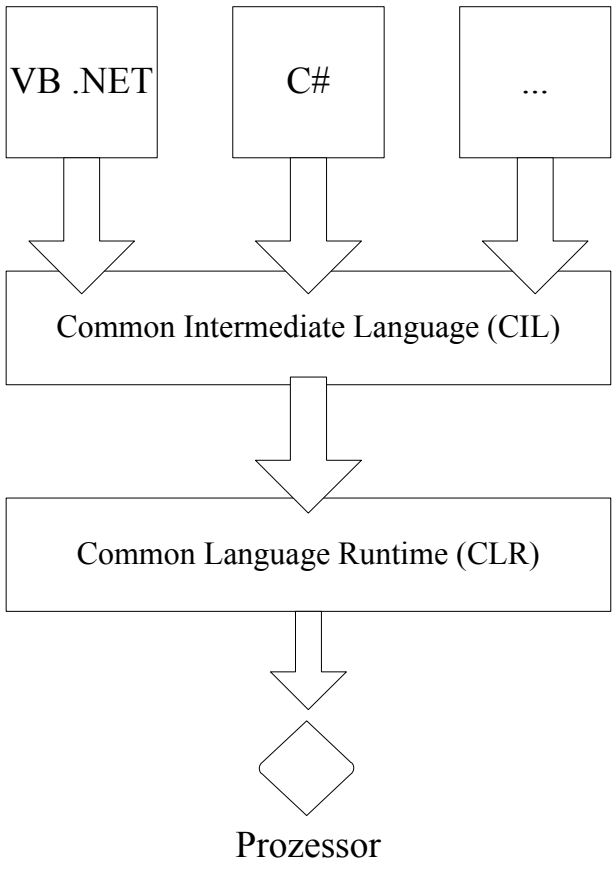

Abbildung 2.2.1-1: Zugrundeliegende Prinzip von .NET

.NET Programme setzten eine Laufzeitumgebung (Common Language Runtime - CLR) voraus. Sie bietet einerseits eine Ablaufumgebung für die Programme, andererseits aber auch Dienste wie Garbage Collection, Sicherheit oder Just-In-Time Übersetzung. Bemerkenswert an der CLR ist, dass sie Programme in allen für .NET angepassten Sprachen ausführen kann. Die CLR führt den standardisierten Zwischencode der CIL

(Common Intermediate Language) aus. Die CIL ist die Übersetzung eines .NET Programms durch einen Compiler. Zur Laufzeit wird dann die CIL in nativen Code für den entsprechenden Prozessor, auf dem das Programm gerade läuft, übersetzt. Da Code aus allen .NET-Sprachen in denselben Zwischencode übersetzt wird, können Funktionen und Klassenbibliotheken, die in verschiedenen .NET-Sprachen geschrieben worden sind, problemlos gemeinsam in einem Programm verwendet werden. Übersetzte Programme (oder Klassen, oder Sammlungen von Objekten, oder …) werden in Assemblies zusammengefasst. Die Programminformationen sind in Modulen, in CIL-kompilierter Form, abgelegt. Es kann mehrere Module pro Assembly geben. In diesem Fall bildet ein Manifest das Inhaltsverzeichnis. Zu jedem Modul kann es Metainformationen geben, die aus dem Quelltext oder Kommentaren ausgelesen werden. Assemblies liegen typischerweise in Form von .exe oder .dll Dateien vor. Aufgrund der enthaltenen Metainformationen ist es in der Regel nicht mehr nötig, die umständliche und fehleranfällige Registrierung der .dll-Datei vorzunehmen.

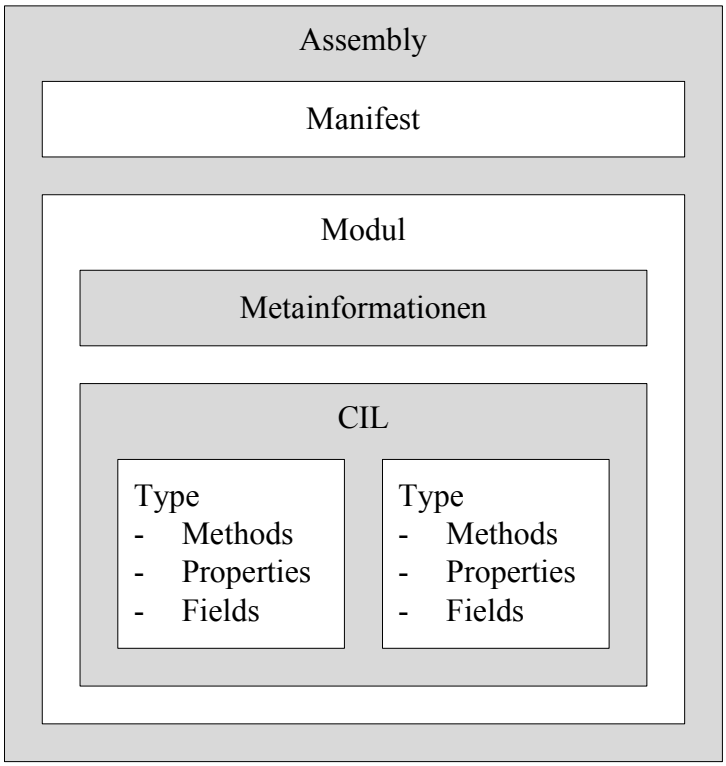

Abbildung 2.2.1-2: Struktur einer .NET Assembly

Die BCL umfasst einige tausend Klassen. Diese sind in Namespaces (Namensräume) unterteilt, um eine gewisse Übersicht durch hierachische Organisation zu gewährleisten.

Die Klassen stellen grundlegende sowie weiterführende Funktionen bereit, die von Programmieren häufig genutzt werden. Eine Dokumentation gibt es in Form eines SDK's (Software Development Kit), sowie online bei MSDN [MSDNDot1]. [WikipNET] [Eber2003]

2.2.2 COM

Das Component Object Model (COM) wurde entwickelt, um unter Windows Interprozesskommunikation und dynamische Objekterzeugung zu ermöglichen. COM-fähige Objekte sind sprachunabhängig und können sowohl DLL's (Dynamic Link Library) als auch ausführbare Programme sein. Jede COM-Komponente bietet mindestens ein Interface an. Dieses kann nach erfolgreicher Instanziierung einer COM-Komponente dazu verwendet werden, die angebotenen Funktionen dieser COM-Komponente einzusetzen. Damit soll COM eine leichte Wiederverwendung von bereits geschriebenem Programmcode möglich machen. COM basiert auf dem Client/Server-Prinzip. Ein COM-Client instanziiert eine COM-Komponente in einem COM-Server und nutzt die Funktionalität des Objektes über Interfaces. Ein COM-Server ist eine DLL oder eine .exe-Datei, welche in einer COM-fähigen Programmiersprache erstellt wurde und COM-Komponenten anbieten sowie erstellen kann. Das COM-Interface dient der Kommunikation vom Client zum Server. Jedes Interface hat eine eindeutige Identifikationsnummer, die GUID (Globally Unique Identifier). Dadurch können auch mehrere Interfaces mit demselben Namen existieren. Die COM-Komponenten selbst haben ebenfalls eine GUID. Mit dieser werden sie in der Windows-Registry eingetragen, damit ein eindeutiger Zugriff auf sie erfolgen kann. COM muss mitgeteilt werden, dass eine entsprechende Komponente threadsicher ist, ansonsten wird COM nur einen Aufruf gleichzeitig an ein Objekt zulassen. Threadsichere Komponenten können hingegen auf jedem Objekt beliebig viele Aufrufe gleichzeitig ausführen. [WikipCOM] [MicrCOM]

2.2.3 Gadgets

Windows Vista Gadgets, sogenannte Minianwendungen, sind im Wesentlichen kleine Webseiten, die mit HTML, XML und JavaScript erstellt werden. Mit Hilfe des Gadget-Objektmodells [MSDNGadg], COM-Komponenten (z.B. ActiveX) (Component Object

Model) ([MicrActX]) oder WMI (Windows Management Instrumentations) können sie auch auf die Konfiguration des Computers zugreifen, Einstellungen anpassen oder Informationen übersichtlich zusammenstellen.

Gadgets werden durch die Windows Vista Sidebar ausgeführt und dargestellt. Von dort kann man die Gadgtes an beliebige Positionen auf dem Desktop platzieren. Hierdurch kann sich die Darstellungsfläche vergrößern, welche bei Ausführung auf der Sidebar begrenzt ist (siehe Abbildung 2.2.3-1 und Abbildung 2.2.3-2). So können zusätzliche Inhalte angezeigt werden. Inhalte, die nur bei Bedarf angezeigt werden sollen, können in einem Flyout (siehe Abbildung 2.2.3-4) untergebracht werden, welches die Informationen bei bestimmten Ereignissen darstellt. Es ist auch möglich, die Gadgets immer im Vordergrund der Fenster zu halten sowie sie (teil-)transparent erscheinen zu lassen. Folgend sind einige Beispielgadgets und ihre Funktionalität abgebildet. [TechGadg]

Abbildung 2.2.3-1: Gadgets abgedockt auf dem Desktop

Abbildung 2.2.3-2: Gadgets angedockt auf der Sidebar

Abbildung 2.2.3-3: Gadget ohne Flyout

Abbildung 2.2.3-4: Gadget mit Flyout

3 Analyse

3.1 Beschreibung der angestrebten Lösung

Die Basis des Systems bildet zum einen Windows Vista mit der Sidebar. Auf dieser lassen sich Gadgets, als Schnittstelle zum Benutzer, installieren. Zum anderen werden Web Services eingesetzt, um die Kommunikation zwischen dem Gadget und einer Datenbanken mit den benötigten Informationen, bereitzustellen.
Da unternehmensweit, wie unter 1.3.1 erwähnt, ausschließlich Microsoft Produkte eingesetzt werden, muss die Lösung in bestehende Strukturen auf Microsoft Basis integriert werden. Die Entwicklung aller Komponenten, bis auf das Gadget, findet auf Basis des .NET Framework 2.0 statt.
Das Hauptziel der angestrebten Lösung ist es, pro-aktiv auf Probleme reagieren zu können. Wenn der Anwender ein Problem meldet, ist dieses schon eingetreten und es kostet Zeit und damit auch Geld, es zu beheben. Insofern wird ein System benötigt, das den Anwender dazu bewegt, selbst etwas zu unternehmen, Probleme zu vermeiden. Die Lösung muss mit möglichst wenig Aufwand für den TÜV-Nutzer verbunden sein, damit dieser sie annimmt. Ebenso muss sie ihm nützlich erscheinen. Aus diesem Grund werden eine Uhr sowie Termine aus Outlook auf einem Gadget angezeigt. Hauptbestandteil wird allerdings ein Tacho sein, dessen Anzeige dem TÜV-Nutzer signalisiert, ob er agieren muss, um Probleme zu vermeiden. In die Anzeige des Flyout's werden weiterhin allgemeine Informationen sowie der Bearbeitungsstatus von eventuell aufgegebenen Problemen bei der ServiceLine und eine detaillierte Anzeige von Outlook Terminen integriert.

Abbildung 3.1-1: Gadget ohne Flyout und Outlook Termine

Beginn	Ende	Betreff
04:00, 3 Feb	19:00, 5 Feb	mehrere tage
AllDayEvent, 4 Feb		ganztägig
AllDayEvent, 4 Feb		ganztägig_2
08:30	11:30	abwesend_1
14:00	15:00	mit vorbehalt
17:30	21:00	mit Vorbehalt_1
20:00	22:00	abwesend_2
22:00, 4 Feb	05:00, 5 Feb	test fuer tagesanzeige
23:00, 4 Feb	09:00, 5 Feb	gestern

Abbildung 3.1-2: Gadget mit Flyout und entsprechenden Outlook Terminen

Die Position der Tachonadel wird durch Berechnung verschiedener Werte bestimmt. Der TÜV-Nutzer kann sich anzeigen lassen, worauf die Position des Zeigers beruht und im Problemfall wird ihm eine Lösung angeboten. Für die Berechnung des Wertes des Tachos werden lokale Informationen vom Client sowie von verschiedenen Datenbanken herangezogen. Abhängig von dem Vergleich (im folgenden Tacho-Vergleich genannt) dieser Daten wird ein Wert gebildet und Informationen bereitgestellt. Diese kann sich der TÜV-Nutzer anzeigen lassen. Der Wertebereich des Tachos reicht von 0 (grün) bis 60 (rot). 0 ist sehr gut, 60 am schlechtesten, demnach ist 30 die Mittelposition des Zeigers. Sobald ein ermittelter Wert den Alarmbereich überschreitet (größer 50), wird die gesamte Anzeige, außer den Zeigern, das Logo und der Tacho, rot eingefärbt (siehe Abbildung 3.1-3). Des Weiteren gibt es einen Wert in der Registry, der dafür sorgt, die gesamte Anzeige orange einzufärben (siehe Abbildung 3.1-4). Diese Option ist für noch nicht bestimmte, besondere Ereignisse vorgesehen. Der ermittelte Wert des Tachos sowie alle Informationen über sein Zustandekommen werden in eine Datenbank geschrieben.

Abbildung 3.1-3: Gadget - Alarmbereich

Abbildung 3.1-4: Gadget - Registry Value

Der Vorteil dieser Lösung für den TÜV-Nutzer liegt in der Möglichkeit, pro-aktiv Probleme zu erkennen und eventuell selbst Gegenmaßnahmen einzuleiten oder zumindest das Problem bei der ServiceLine zu melden. Dadurch wird ein Ausfall von Arbeitszeit vermieden und der TÜV-Nutzer kann mit einem verlässlichen Gerät arbeiten.

Hinzu kommt die übersichtliche Zusammenfassung vieler relevanter Informationen in dieser Anwendung. Der TÜV-Nutzer kann auf einen Blick mit wenigen Mausklicks den Status seines Gerätes erkennen, den Fortschritt seiner bei der ServiceLine eingereichten Probleme abrufen, alle Outlook Termine des aktuellen Tages ansehen, die Uhrzeit ablesen sowie andere allgemein wichtige Informationen für die Belegschaft lesen.

Für die Administratoren ergibt sich der Vorteil, Geräte mit bevorstehenden Problemen direkt erkennen zu können. Sie können sich mit den TÜV-Nutzern in Verbindung setzen und dabei helfen, die Probleme nicht entstehen zu lassen. Hinzu kommt, dass in Statistiken ersichtlich gemacht werden kann, wie sich spezielle Änderungen an den Clients in der Gesamtheit aller Clients auswirken, positiv sowie negativ.

Das Gadget benötigt für bestimmte Ereignisse nicht zwangsläufig eine Verbindung zum TÜV Netzwerk, um vor bevorstehenden Problemen zu warnen. Die Daten des SCOM Client zur Festplattenüberwachung können durch das Gadget ausgewertet werden. Es können Informationen auf dem Client hinterlegt werden, zu welchem Zeitpunkt zum Beispiel eine Preisanpassung stattfindet. Diese Information kann durch das Gadget ausgewertet werden und es macht den TÜV-Nutzer darauf aufmerksam, dass er einen Abgleich mit den TÜV Systemen durchführen muss, um die aktuellen Preislisten zu bekommen. Das Gadget kann darauf aufmerksam machen, dass generell lange keine Verbindung mehr mit dem TÜV Netz bestand, zwecks Abgleich und/oder Aktualisierung von Daten.

Um zu gewährleisten, dass dem TÜV-Nutzer aktuelle Daten seines Clients angezeigt werden, kann er einen Datenabgleich zwischen seinem Gerät und den entsprechenden Referenzen aus einer Datenbank anstoßen. Außerdem wird in einem definierten Zeitabstand vom Gadget ermittelt, ob eine Verbindung zum TÜV Netz besteht, um aktuelle Daten abzurufen.

Es wird eine Gewichtung der Ergebnisse des Tacho-Vergleichs der Client- mit den Referenzdaten geben. So können die Ergebnisse differenzierter in das Endergebnis

einfließen und es gibt die Möglichkeit, Resultate unabhängig voneinander zu bewerten, positiv als auch negativ.

Um unterschiedliche Gruppen von Clients individuell mit entsprechenden Daten versorgen zu können, werden alle Clients in verschiedene Gruppen unterteilt. So ist gewährleistet, dass individuelle Merkmale von Gruppen mit berücksichtigt werden können und andererseits alle Gruppen nur die Informationen zur Auswertung bekommen, die auch für sie bestimmt sind.

Um diese Lösung zu realisieren, wird auf dem Client eine COM-Komponente installiert, die sich mit einem Web Service verbindet. Dieser wiederum nimmt Kontakt mit einer Datenbank auf, um die für den Client erforderlichen Daten abzurufen. Der Web Service schickt die Daten an den Client, damit dieser mit der Auswertung beginnen kann. Ist diese abgeschlossen, wird ein Wert entsprechend des Ergebnisses in der Registry gesetzt, der vom Gadget ausgelesen wird und die Position des Zeigers auf dem Tacho bestimmt. (siehe Abbildung 3.1-5)

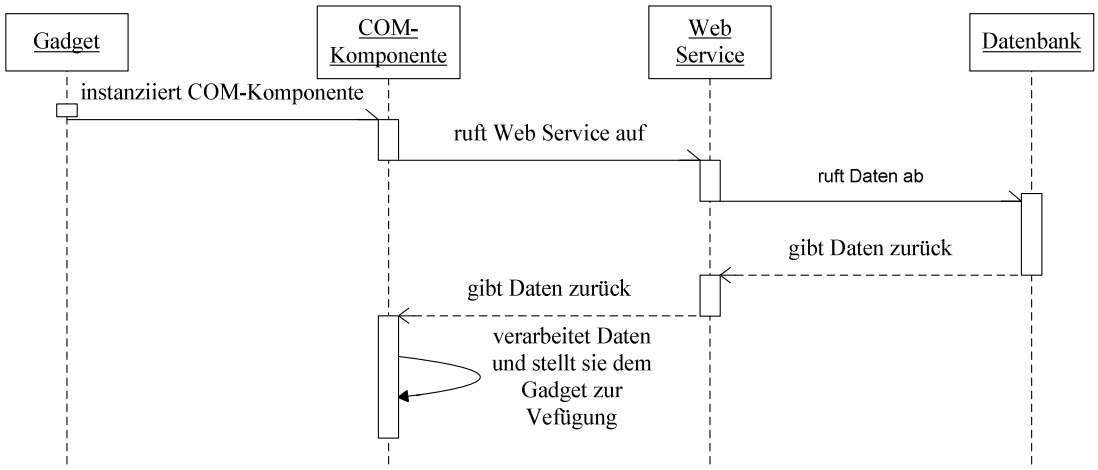

Abbildung 3.1-5: Darstellung der Abläufe

Für die Datenbank wird eine GUI entwickelt, um die relevanten Daten komfortabel eingeben zu können. Außerdem wird eine Prüfung der eingegebenen Daten erfolgen, um sicherzustellen, dass diese den Konventionen der COM-Komponente entsprechen und ohne Fehler verarbeitet werden können.

3.2 Realisierungskonzept – Pflichtenheft

Die Anforderungen, welche an das Projekt gestellt werden, wurden über einen längeren Zeitraum erarbeitet und auch während des laufenden Projektes ergaben sich durch Prototypen und die darauf folgenden Gespräche mit Kollegen noch weitere Anforderungen (siehe 3.2.3). Insofern ist die Festsetzung der Anforderungen noch nicht abgeschlossen. Einige grundsätzliche Forderungen wurden allerdings zu Beginn des Projektes durch den Projektleiter (gleichzeitig auch Kunde) vorgegeben. Hierzu zählen die Informationen, die auf dem Gadget (nicht dem Flyout) dargestellt werden, das Erscheinungsbild des Gadgets (eine Uhr mit TÜV Logo und integrierten Tacho), eine Alarmfunktion ab einem bestimmten Tacho-Schwellwert für das Gadget, die Integration der Zuverlässigkeitsüberwachung von Windows Vista, die Anzeige von Outlook-Terminen sowie die Möglichkeit des Inventarisierens des erzeugten Tachowertes über SCOM. Für die Kommunikation mit den Datenbanken soll eine COM-Komponente für den Client erstellt werden, welche die Verbindung zu einem Web Service herstellt. Dieser wiederum sorgt für die Bereitstellung der Daten aus den Datenbanken auf dem Client.
Die folgende Einteilung der Anforderungen erfolgt nach den Kriterien der ISO/IEC 9126 [ISOIEC91] [WikipIS1]. In der ersten Spalte wird ein Index eingeführt, der die Bezugnahme aus anderen Kapiteln erleichtert und darauf hinweist, dass es sich um eine funktionale (Fx) beziehungsweise nicht Funktionale (nFxx) Anforderung handelt. In der zweiten Spalte wird die Priorität der Anforderung angegeben: 1 - must-have, 2 - nice-to-have. Die dritte Spalte gibt die Anforderung wieder.

3.2.1 Funktionale Anforderungen

Index	Priorität	Anforderung
F1	1	Die Erstellung eines Windows Vista Gadgets, zur Visualisierung der Ergebnisse der Tacho-Vergleiche und zur Hilfestellung bei dadurch entdeckten Problemen für den TÜV-Nutzer.

F2	1	Die Erstellung einer COM-Komponente für den Client des TÜV-Nutzers. Diese übernimmt die Kommunikation mit dem Web Service sowie die Auswertung der von ihm gelieferten Daten.
F3	1	Die Erstellung eines Web Services für die Kommunikation zwischen Datenbank und Client.
F4	1	Die Erstellung einer Datenbank (im folgenden Tacho-Vergleich-Datenbank genannt), in welcher alle Tacho-Vergleiche hinterlegt sind. Sie enthält eine Standardtabelle, welche für alle Clients gültig ist. Des Weiteren mehrere Spezialtabellen, die nur für spezielle Clientgruppen Gültigkeit haben.
F5	1	Es wird die Zuverlässigkeitsüberwachung von Windows Vista ausgelesen und mit in die Tacho-Vergleiche einbezogen.
F6	1	Der erzeugt Tachowert entspricht einer Konvention, so dass er durch SCOM inventarisiert werden kann
F7	2	Für die Tacho-Vergleiche relevante Referenzwerte werden aus der HPOpenView-Datenbank bezogen.
F8	2	Für die Tacho-Vergleiche relevante Referenzwerte werden aus der SCOM-Datenbank bezogen.
F9	2	Eine Gewichtung der Ergebnisse der Tacho-Vergleiche wird durchgeführt.
F10	2	Die Erstellung einer Datenbank, in welcher alle Ergebnisse der Tacho-Vergleiche hinterlegt sind.
F11	2	Es muss eine Unterscheidung der Clients nach verschiedenen Gruppen möglich sein, da unterschiedliche Gruppen von Clients unterschiedliche Tacho-Vergleiche nötig machen.
F12	2	Um die übertragenen Datenmengen zwischen Client und Web Service auf ein Minimum zu reduzieren sowie die Belastung auf dem Client durch die Auswertung möglichst gering zu halten, werden nur diejenigen Daten übertragen, die sich seit der letzten Übertragung geändert haben oder generell gefordert sind.

F13	2	Alle Fehlermeldungen, die im beschriebenen System auftreten, werden in das Windows EventLog geschrieben, um eine Erfassung durch SCOM zu gewährleisten
F14	2	Auf dem Client müssen die Möglichkeit der Ausführung eines beliebigen Programms und die darauf folgende Auswertung des gelieferten ReturnCodes gewährleistet sein.
F15	2	Eine Authentifizierung der Clients gegenüber dem Web Service muss gewährleistet sein.
F16	2	Die Authentifizierung der Clients gegenüber dem Web Service erfolgt durch Computerzertifikate.

Die Anforderung F5 kann erst erfüllt werden, wenn von Microsoft eine Schnittstelle zum Ansprechen der Zuverlässigkeitsüberwachung bereitgestellt wird. Dies soll nach Informationen des Projektleiters mit dem Service Pack 1 für Windows Vista geschehen.

3.2.2 Nicht funktionale Anforderungen

Index	Priorität	Anforderung
Zuverlässigkeit:		
nFz1	2	Änderungen an den Datensätzen der Tacho-Vergleich-Datenbank werden zu Testzwecken zuerst einer ausgewählten Gruppe von Clients bereitgestellt, um eventuelle Fehler erkennen zu können, bevor die Änderungen für die Allgemeinheit freigeschaltet werden.
nFz2	2	Die Erreichbarkeit des Web Services durch die Clients muss insofern gesichert sein, als das mindestens einmal am Tag eine erfolgreiche Verbindung mit dem Web Services zustande kommen kann.
Benutzbarkeit:		
nFb1	1	Das Gadget beinhaltet die Funktion eines Tachos.
nFb2	1	Der Tacho des Gadgets dient der Visualisierung von vorhandenen Client Problemen.

nFb3	1	Der Tacho des Gadgets wird in das Logo des TÜVs integriert.
nFb4	1	Das Gadget weist eine Funktion auf, mit der die gesamte Oberfläche rot eingefärbt wird.
nFb5	1	Das Gadget weist eine Funktion auf, mit der die gesamte Oberfläche orange eingefärbt wird.
nFb6	1	Das Gadget hat die Funktion einer Uhr.
nFb7	1	Das Gadget stellt Outlook Termine dar.
nFb8	1	Die Termine aus Outlook werden auf der Uhr ab der aktuellen Uhrzeit für die folgenden elf Stunden dargestellt, damit es nicht zu einer Überlappung der angezeigten Termine kommt.
nFb9	1	Die Darstellung der Termine aus Outlook auf der Uhr funktioniert auch über Datumsgrenzen hinweg.
nFb10	1	Die Darstellung der Termine aus Outlook auf der Uhr erfolgt priorisiert nach der Outlook-Option “Anzeigen als“. Die höchste Priorität wird bei Konflikten zur Anzeige gebracht.
nFb11	1	Die Priorität aus der Outlook-Option “Anzeigen als“ ergibt sich wie folgt (0 = niedrigste Priorität, 3 = höchste Priorität): “Frei“ = 0, “mit Vorbehalt“ = 1, “Gebucht“ = 2, “Abwesend“ = 3.
nFb12	2	Auf dem Flyout werden alle Termine des aktuellen, des vorherigen und des folgenden Tages aus Outlook dargestellt.
nFb13	2	Die Darstellung von Informationen über erkannte Probleme und entsprechende Lösungsvorschläge auf dem Flyout.
nFb14	2	Die Darstellung von aufgegebenen Problemen bei der Service Line und deren Bearbeitungsstatus auf dem Flyout.
nFb15	2	Die Darstellung von allgemeinen Informationen für die Belegschaft auf dem Flyout.
nFb16	2	Das Gadget ist multilingual.
nFb17	2	Die statistische Darstellung der gesammelten Daten der Clients in Form einer Fieberkurve.

nFb18	2	Die Möglichkeit, sich die Tacho-Vergleiche von zehn Clients detailliert anzeigen zu lassen, deren Ergebnisse der Auswertung der Tacho-Vergleiche am schlechtesten ausgefallen sind.

Effizienz:

nFe1	1	Der Abruf der Daten darf nicht länger als 5 Sekunden dauern.
nFe2	2	Das Gadget behält die volle Funktionalität, während im Hintergrund Aufgaben bearbeitet werden.

Änderbarkeit:

nFä1	2	Änderungen an den Datensätzen der Tacho-Vergleich-Datenbank dürfen keine Änderungen am Web Service und am Client nötig machen.

Übertragbarkeit:

An die Übertragbarkeit werden keine Anforderungen gestellt. Aufgrund der Realisierung auf Grundlage von Microsoft Produkten ist die Portierung allerdings auf diese Systeme beschränkt. Außerdem ist das Produkt an die Anforderungen des TÜV zugeschnitten, insofern nur bedingt in anderen Umgebungen einsetzbar.

3.2.3 Änderungen der Anforderungen während des Projektes

Die unter 3.1.1 und 3.1.2 genannten Anforderungen gelten für den dritten Prototyp. Sie haben sich im Verlauf der Arbeit entwickelt. Folgende Anforderungen galten für den ersten Prototypen: F1, F2, F3, F5, F6, … , F9, F15, nFb1, … , nFb13, nFe1, nFe2, nFä1. Im Zuge der Entwicklung des ersten Prototypen haben sich weitere Anforderungen ergeben: F12, F15, F16, nFb14. Durch die Entwicklung des zweiten Prototypen sind folgende Anforderungen hinzugekommen: F4, F10, F11, F12, F14, nFz1, nFz2, nFb15, … , nFb18.

Anhand der oben beschriebenen Entwicklung der Anforderungen ist es möglich ein System zu implementieren, das den Wünschen des Kunden entspricht. Hätte man ein System nach den Anforderungen des ersten Prototyps erstellt, so wäre der Kunde damit aller Wahrscheinlichkeit nach höchst unzufrieden gewesen.

3.3 Szenarien für mögliche Realisierungen

3.3.1 ActiveX

ActiveX bezeichnet ein Softwarekomponenten-Modell von Microsoft für aktive Inhalte unter Windows. ActiveX-Komponenten erweitern die COM-Standards von Microsoft. Es sind Softwarekomponenten für andere Anwendungen. Sie können in verschiedenen Programmiersprachen und Umgebungen verwendet werden. Einige Programme nutzen zum Beispiel den Internet Explorer über ActiveX zur Anzeige von Informationen. [WikipAct]

Windows Vista nutzt die Sidebar um Gadgets zur Ausführung zu bringen. Die Sidebar wiederum nutzt die Rendering Engine des Internet Explorers um die Gadgets darzustellen. Da die Gadgets mittels JavaScript und HTML programmiert beziehungsweise dargestellt werden und im Sicherheitskontext des Internet Explorers laufen, müssen sie auf beispielsweise ActiveX-Objekte[MSDNAct2] zurückgreifen, um auf den Client, auf dem sie laufen, zugreifen zu können.

Unter Windows Vista lassen sich ActiveX-Objekte nicht mehr ohne weiteres installieren. Beim TÜV NORD wird über eine Group-Policy geregelt, welche Bedingungen erfüllt sein müssen, damit ein Objekt installiert werden darf: Es muss ersichtlich sein, woher das Objekt stammt und es muss ein gültiges Zertifikat aufweisen [TECHDerA].

3.3.2 .NET COM-Komponente

Mit .NET ist es möglich Komponeneten zu implementieren, die sich wie COM-Komponenten verhalten. Damit sind sie für Anwendungen wie COM-Komponenten ansprechbar. Für diese .NET COM-Komponenten gelten die Group-Policy Einstellungen, wie sie für ActiveX gültig sind, nicht. Da sie aber wie COM-Komponenten ansprechbar sind (sobald sie als COM-Objekte registriert sind), können sie auch von Anwendungen instanziiert werden, die auf ActiveX-Komponenten angewiesen sind. [MSDNAct1]

3.3.3 WSE - Router

Der WSE (Web Service Enhancements)-Router dient als Vermittlungsstelle zwischen den Nutzern von Web Services und den Web Services selbst. Er selbst ist auch ein Web Service. Er vermittelt eingehende Anforderungen an die entsprechenden Server und überprüft, ob die Nutzer berechtigt sind, die Dienste zu nutzen. Dies kann mittels X.509-Zertifikaten, UsernameToken, UsernameOverTransport und AnonymousOverX509 erfolgen. Diese Verfahren werden im Folgenden beschrieben.
Ein X.509-Zertifikat bestätigt, dass die Person, die die Anforderung sendet, hierzu berechtigt ist. Das Zertifikat kann von der hauseigenen Zertifizierungsstelle (PKI – Public Key Infrastructure) ausgestellt werden. Zusätzlich zum X.509-Zertifikat können folgende Maßnahmen ergriffen werden: Das UsernameToken verwendet eine Kombination aus Benutzername und Kennwort, die in einer externen Datenbank, zum Beispiel dem Active Directory, gespeichert sind. Da ein Kennwort übertragen wird, muss dafür Sorge getragen werden, dass die Kommunikation verschlüsselt erfolgt. Diese beiden Verfahren übertragen die Anmeldeinformationen auf Nachrichtenebene, in der SOAP-Nachricht. UsernameOverTransport verwendet ebenfalls eine Kombination aus Benutzername und Kennwort zur Authentifizierung. Allerdings werden diese Informationen per SSL (Secure Sockets Layer) übertragen, also über die Transportschicht. Bei AnonymousOverX509 (oder auch AnonymousForCertificate) authentifiziert sich der Nutzer mit dem öffentlichen Schlüssel des Servers. Daraus folgt, dass der Client dem Server unbekannt ist. Also sollte dieses Verfahren nur bei unkritischen Diensten oder einem bekannten eingeschränkten Teilnehmerkreis angewandt werden.
Durch den WSE-Router ist es möglich, die hinter ihm liegende Netzwerktopologie für die Nutzer zu verbergen. Ebenso ist möglich, die Server der Web Services zu warten, ohne das es für die Nutzer zu Ausfallzeiten kommt. Durch eine Konfigurationsänderung am WSE-Router werden die eingehenden Anforderungen an einen alternativen Server umgeleitet, bis die Wartungsarbeiten beendet sind und der ursprüngliche Server wieder verfügbar ist. [Morg2007]

3.3.4 UDDI

UDDI ist ein Verzeichnisdienst wie in 2.1.3 beschrieben. Es besteht bei UDDI die Möglichkeit, mittels X.509-Zertifikaten eine Authentisierung durchzuführen. Es ist ebenso möglich, eine Verbindung mittels SSL zu verschlüsseln [Schi2007]. Eine Routing-Funktionalität wird durch UDDI nicht unterstützt. Es ist allerdings möglich, den Nutzer/Client eines Web Services so zu implementieren, dass dieser eine Anfrage beim UDDI Dienst stellt, sobald er den gewünschten Web Service nicht mehr erreicht. In diesem Fall wird über den UDDI Dienst versucht, einen äquivalenten Web Service zu dem ausgefallenen zu finden. Ist dies möglich, so wird durch den Web Service Nutzer/Client eine Verbindung zu dem neuen Web Service aufgebaut.

4 Realisierung

Als Entwicklungsumgebung kommt Visual Studio 2008 zum Einsatz, da es exzellente Möglichkeiten für die Entwicklung von .NET basierten Anwendungen und ebenso Möglichkeiten für die JavaScript Entwicklung bietet. Das Gadget wurde in JavaScript programmiert und wird mittels HTML dargestellt. Die COM-Komponente sowie der Web Service wurden in C# entwickelt.

Aus zeitlichen Gründen kann keine vollständige Umsetzung der Anforderungen erfolgen. Die folgenden Unter-Kapitel beschreiben daher den Stand der Entwicklung zum jeweiligen Zeitpunkt auf prototypischer Basis. Weiterhin wird in den Kapiteln 4.2 und 4.3 darauf eingegangen, welche Anforderungen die jeweiligen Prototypen verfehlt haben und weshalb als Konsequenz weitere Prototypen entwickelt werden mussten. Der dritte Prototyp beschreibt den aktuellen Entwicklungsstand.

4.1 Das Gadget

Vor Beginn der Programmierung der folgenden drei Prototypen ist das Gadget (siehe 3.1 + 2.2.3), in dem Umfang, wie es dieser Arbeit zugrunde liegt, bereits von dem Verfasser dieser Arbeit fertig gestellt worden. Die Prototypen umfassen also nur die Entwicklung der Infrastruktur für die Bereitstellung der Daten, die das Gadget benötigt.
Die Funktionalität des Gadgets wird durch JavaScript realisiert. Allerdings werden alle Funktionen, die den Tacho betreffen, von der COM-Komponente ausgeführt. So ist das Gadget für die Outlook Termine, die Uhr sowie das instanziieren der COM-Komponente und die dynamische Darstellung zuständig.
Das Ergebnis der Tacho-Vergleiche der COM-Komponente wird in einen Registry-Schlüssel geschrieben. Das Gadget führt in einem bestimmten Zeitabstand regelmäßig die Funktion `GetIndex()` der COM-Komponente aus, die dem Gadget den Ergebniswert zurückliefert.
Während der Programmausführung schreibt das Gadget zu Debuggingzwecken ein Log. Dieses liegt innerhalb der Verzeichnisstruktur des Gadgets und wird nicht größer als zwei MegaByte.
Im Installationspaket des Gadgets sind alle relevanten Dateien des Gadgets vorhanden. Es lässt sich mit einem Packprogramm, das das zip-Verfahren beherrscht, entpacken. Die folgenden .js-Dateien enthalten die Logik des Gadgets, alle weiteren Dateien im Installationspaket dienen der Darstellung des Gadgets.

- tacho.js : Uhr und Tacho
- outlook.js : Outlook-Termine
- flyout.js : Darstellung des Flyout
- tacho_outlook.gadget : Installationspaket des Gadgets

4.1.1 Outlook

Der Zugriff auf Outlook findet über die Scripting-Schnittstellen von Outlook [MSDNOut1] durch den Windows Script Host statt. In der Funktion `getAppointments()` werden über `outlookFolder.Items()` alle Termine aus Outlook abgerufen. Danach werden durch einen `filter` und `objItems.restrict(filter)` alle

nicht benötigten Termine ausgeschlossen. Die Termine werden in ein dreidimensionales Array `items` einsortiert. Diejenigen mit der niedrigsten Priorität in [0][x][x], die mit der höchsten in [3][x][x]. Jeweils ein Termin-Item (Termin + Attribute) kann mit [x][x] abgerufen werden. Zugriff auf die Attribute erhält man durch [x][x][x] (siehe Abbildung 4.1.1-1). Das `items`-Array enthält die Termine des aktuellen, des vorherigen sowie des folgenden Tages. Ebenfalls wird die Anzeige der Termine auf dem Flyout durch den String `flyoutContent` vorbereitet.

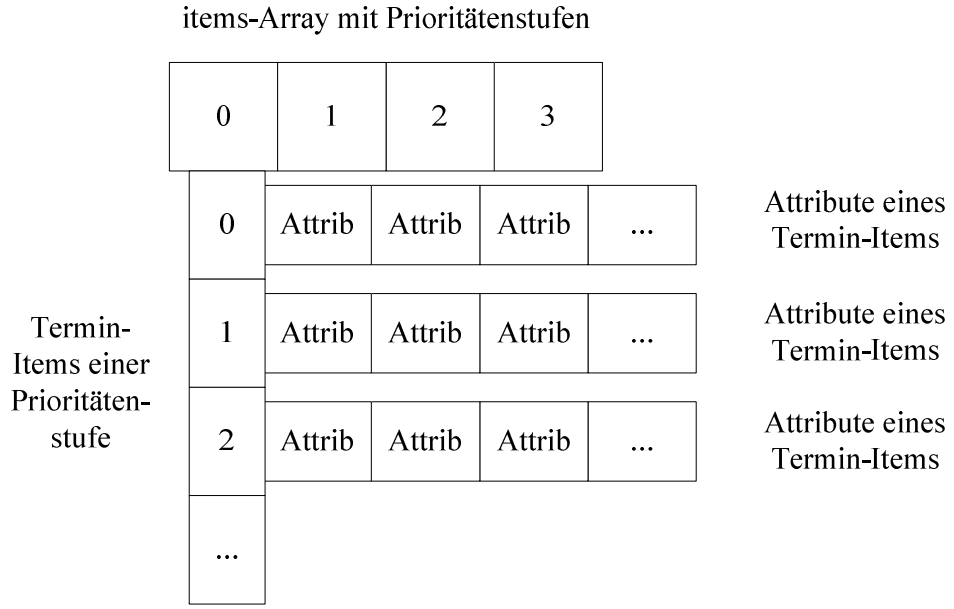

Abbildung 4.1.1-1: Aufbau des items-Array

In der Funktion `setAppointmentPictures()` wird durch das `items`-Array iteriert, um die auf der Uhr anzuzeigenden Termine herauszufiltern. Hierfür wird ein weiteres Feld im `items`-Array benötigt (`items[busyStatiArray]`), welches wiederum ein Array ist und 48 Felder hat: für jede halbe Stunde des Tages eins. Nach Prüfung aller Kriterien des erfassten Termins, ob er zum aktuellen Zeitpunkt angezeigt werden darf, erfolgt ein Eintrag in das entsprechende Feld des Arrays gemäß seiner Priorität oder auch nicht.
Nun werden durch die Funktion `setOLClockPics()` die Felder aus `items[busyStatiArray]` ausgewertet. Entsprechend dem Prioritäteneintrag wird eine Farbe ausgewählt (1: hellblau; 2: blau; 3: rot) und zur passenden Uhrzeit auf der Uhr des Gadgets angezeigt.

4.1.2 Uhr

Die aktuelle Uhrzeit wird sekundengenau aus der Systemzeit durch die Funktion `getTime()` ermittelt. Zugriff auf die Systemzeit wird über den Windows Script Host erlangt. Für die Position der Uhrzeiger wird dann eine Umrechnung der aktuellen Uhrzeit in Winkel durch `calculateTime()` vorgenommen, da es über die Gadget API [MSDNGadg] möglich ist, ein Bild um einen Punkt in 360 Grad Schritten rotieren zu lassen. Dieser Vorgang wiederholt sich alle 0,999 Sekunden, damit ein maximaler Zeitversatz zwischen Systemzeit und angezeigter Zeit von einer Sekunde entsteht und die Systemleistung Aufgrund der Abfrage der Systemzeit durch pollen nicht zu sehr in Mitleidenschaft gezogen wird.

4.1.3 Dynamische Darstellung

Durch die Methode `analyzeValues()` werden alle elf Sekunden die Registrywerte abgerufen, auf welchen die Berechnung der Position des Tacho Zeigers sowie der Anzeige des roten sowie orangenen Hintergrundes beruht. Sollte das Flyout geöffnet sein, werden durch die Methode `addContentToFlyout()` alle relevanten Informationen abgerufen, die zur Darstellung auf dem Flyout benötigt werden. Die Termininformationen aus Outlook auf dem Rand der Uhr werden alle 20 Sekunden durch die Methode `setAppointmentPictures()` aktualisiert.

4.2 Der erste Prototyp

4.2.1 Beschreibung

Der erste Prototyp basiert auf dem Gadget wie in 4.1 beschrieben. Es gibt eine COM-Komponente, die die Kommunikation mit dem Web Service abwickelt. Der Web Service sorgt für die Verbindung zu den Datenbanken und ruft die entsprechenden Informationen ab. Sobald er diese erhalten hat, gibt er sie an die COM-Komponente zurück. Diese wertet die Daten aus und setzt entsprechend einen Registry Wert. Die gesamte Kommunikation ist synchron aufgebaut, wodurch die COM-Komponente sowie der Web Service erst mit ihrer Programmabarbeitung fortfahren können, wenn sie eine Antwort bezüglich ihres Aufrufes bekommen haben (siehe Abbildung 4.2.1-1).

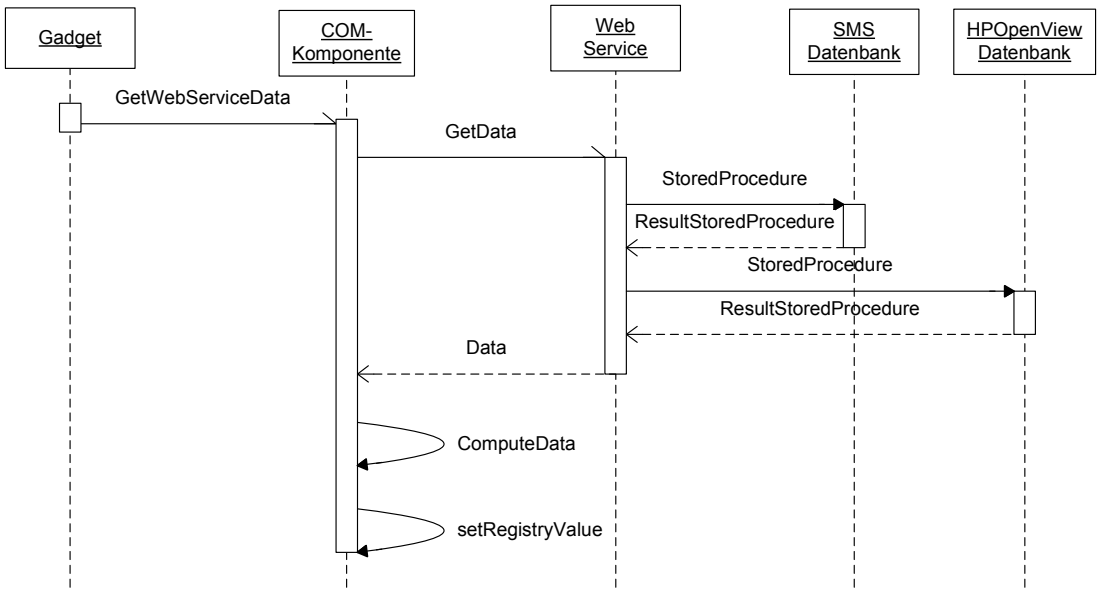

Abbildung 4.2.1-1: Kommunikation der Komponenten des ersten Prototypen

4.2.2 Technische Details

Der erste Prototyp basiert unter anderem auf der Annahme, dass der UDDI-Service eine Art Routing-Funktion übernehmen kann und die Anfragen an Web Services immer zu den nachgefragten Web Services weiterleitet. Diese Funktionalität besitzt er allerdings nicht (siehe 3.3.4). Für die Implementierung des Web Services sind die Funktionen des

UDDI-Services, der für dieses Projekt durch den Autor eingerichtet worden ist, aber genutzt worden.

In diesem Entwurf ist der Performance-Gesichtspunkt nicht ausreichend beachtet worden. Das Design der Anwendung ist synchron aufgebaut worden: Das Gadget ruft die COM-Komponente auf, welche wiederum den Web Service synchron aufruft und dieser wiederum eine Stored Procedure auf einer SCOM- beziehungsweise HPOpenView-Datenbank synchron ausführt. Von diesen Aufrufen wurden testweise 1000 hintereinander ausgeführt. Bis die letzte Antwort der Aufrufe bei der COM-Komponente wieder eingetroffen ist sind im Mittel 6,2 Sekunden vergangen.

Als Rückgabewert des Web Services wurde ein String-Array gewählt. Diese Wahl fiel Aufgrund der Annahme, dass nur Registry Werte verglichen werden müssen. In den ersten Tests war ein String-Array überzeugend. Es wurden nur kleine Datenmengen übertragen und sie bestanden nur aus Strings. Ebenso wird die Serialisierung von Strings für die Übertragung vom Web Service zur COM-Komponente automatisch durchgeführt.

Zur Sicherung der Kommunikation zwischen der COM-Komponente und dem Web Service wird die Verbindung per SSL verschlüsselt. Als Anmeldeinformationen am Web Service kommen die Authentifizierungsdaten des angemeldeten Benutzers zum Tragen.

Die Konfiguration der Verbindung zur Datenbank wurde im Code des Web Services vorgenommen und die Verbindung wird auch von dort initialisiert. Für den Zugriff wurde ein spezieller Benutzer angelegt und mit Leserechten ausgestattet. Zu diesem Zeitpunkt war der Abruf der Informationen aus den Datenbanken über mehrere Stored Procedures vorgesehen. Diese Stored Procedures arbeiten auf der vorhandenen SCOM- sowie HPOpenView-Datenbank.

Der Web Service bereitet die abgerufenen Daten für den Client so auf, dass nur die für ihn bestimmten Daten übermittelt wurden. Diese Daten wurden von der COM-Komponente auf dem Client entgegengenommen, woraufhin die entsprechenden Daten des Clients aus der Registrierung ausgelesen und verglichen wurden. Als Ergebnis des Tacho-Vergleichs wird ein Wert in die Registrierung geschrieben.

4.2.3 Verfehlte Anforderungen

Dieser Prototyp hat die folgenden Anforderungen nicht erfüllt: F13, F16, nFb14, nFe1. Ausschlaggebend war die Verfehlung der Anforderung nFe1. Der Abruf und die Verarbeitung der Daten hat 6,2 Sekunden gedauert. Hierdurch kam es zur Programmierung eines weiteren Prototyps. Die Anforderungen F13, F16, nFb14 sind neue, sie wurden während der Erstellung des ersten Prototyp festgelegt.

Die Anforderung F13 ergab sich durch das Gespräch mit einem Kollegen, wie das implementierte System Fehler protokollieren soll. Da es durch SCOM die Möglichkeit gibt, Einträge des EventLogs zentral einzusammeln und auszuwerten, wurde es als Anforderung übernommen.

Die Anforderung F16 entstand ebenfalls in einem Gespräch mit diesem Kollegen. Nachdem klar war, dass der UDDI Service nicht die benötigten Funktionen für ein Routing zur Verfügung stellt, wurde nach Alternativen gesucht. In diesem Zusammenhang kam auch die Frage auf, wodurch sich ein TÜV-Nutzer dem Web Service gegenüber authentifizieren könne. Da manche Mitarbeiter nur lokal und nicht in der Domäne an ihren Computern angemeldet sind, kann eine Authentifizierung nicht über Nutzerzertifikate, sondern nur über Computerzertifikate erfolgen. Des Weitern analysiert das zu implementierende System Computer, unabhängig von den auf ihnen arbeitenden Nutzern. Insofern muss sich der Computer dem Web Service gegenüber ausweisen. Die Anforderung nFb14 entstand durch den Projektleiter, als mit ihm die aktuelle Entwicklung des Systems besprochen wurde.

4.3 Der zweite Prototyp

4.3.1 Beschreibung

Der zweite Prototyp basiert auf dem Gadget wie in 4.1 beschrieben. Auch hier gibt es eine COM-Komponente, die die Kommunikation mit dem Web Service abwickelt. Der Web Service sorgt für die Verbindung zu den Datenbanken und ruft die entsprechenden Informationen ab. Sobald er diese erhalten hat, gibt er sie an die COM-Komponente zurück. Diese wertet die Daten aus und setzt entsprechend einen Registry Wert. Im Gegensatz zum ersten Prototyp ist die gesamte Kommunikation asynchron aufgebaut (siehe Abbildung 4.3.1-1).

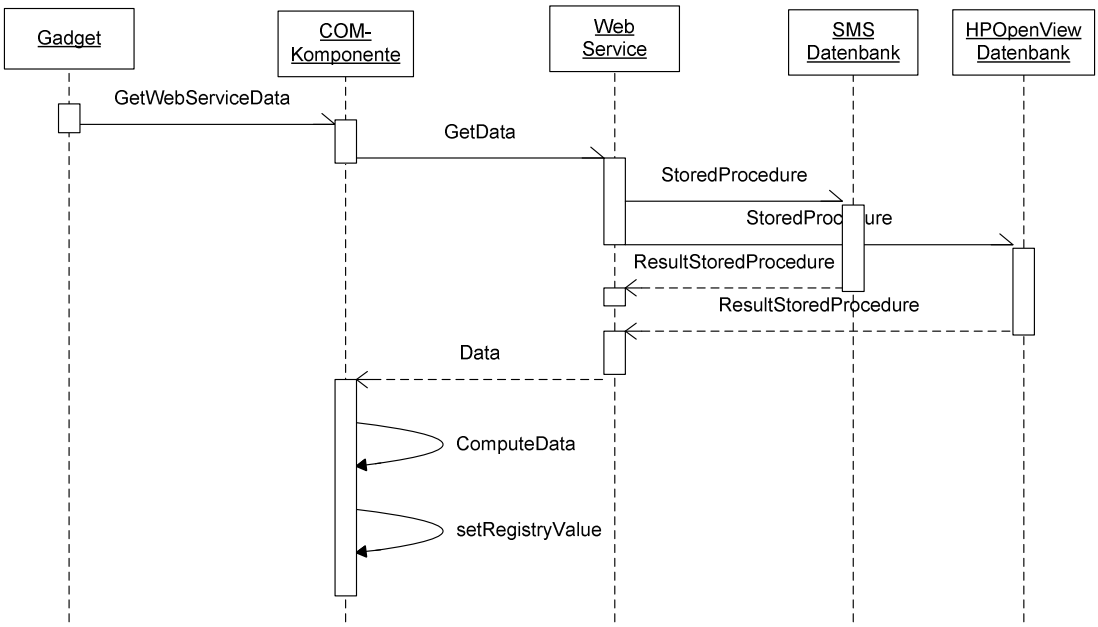

Abbildung 4.3.1-1: Kommunikation der Komponenten des zweiten Prototypen

4.3.2 Technische Details

Die technische Beschreibung dieses Prototyps beschränkt sich auf die Unterschiede zum ersten Prototyp. Beim zweiten Prototyp wurde die Verbindung zwischen COM-Komponente und dem Web Service fest einprogrammiert. Mithilfe von Visual Studio ist

ein Proxy auf Clientseite erstellt worden, der die Adresse des Web Service enthält. Der UDDI-Service ist hier nicht berücksichtigt worden.

Diese Lösung wurde von Beginn an asynchron aufgebaut (siehe Abbildung 4.3.2-1). Visual Studio stellt Methoden zum asynchronen Aufruf von Web Services bereit. Microsoft beschreibt drei Szenarien, wie diese asynchronen Aufrufe umzusetzen sind [Morg2007]. Das erste Szenario wird durch Polling realisiert. Es schied aus, da die beiden anderen Ansätze ohne Polling auskommen. Das zweite basiert auf einem asynchronen Aufruf, der nach Abschluss eine anzugebende Methode aufruft (Callback). Diese Möglichkeit schied aus, da mehrere Aufrufe zu synchronisieren waren, um alle Ergebnisse miteinander auswerten zu können. Dies ist mit diesem Szenario nicht möglich. Das dritte Szenario basiert auf dem Aufruf zweier Methoden: BeginXXX() und EndXXX(), wobei XXX für den Methodennamen der aufzurufenden Methode steht. Mit BeginXXX() wird der asynchrone Aufruf der Methode XXX() gestartet (1). Sobald die Bearbeitung der Methode (2) abgeschlossen ist, wird ein Event gefeuert (3). Die aufgerufenen Methoden haben damit ihr Programm abgearbeitet und stellen nun noch ihre Ergebnisse zum Abruf bereit (4). Durch die Events wird es möglich, ein WaitHandler-Array zu implementieren, diesem die Referenzen aller Events der aufgerufenen Methoden zu übergeben und mit WaitAll() auf das Feuern der Events, das heißt auf die Abarbeitung aller Aufrufe, zu warten. Danach sind sicher alle Rückgabewerte vorhanden und man kann mit der Methode EndXXX () die Ergebnisse der Aufrufe abrufen (5 und 6). Danach kann die Auswertung beginnen beziehungsweise die Programmabarbeitung fortgesetzt werden (7). Allerdings wird dieses Szenario seit Visual Studio 2003 beziehungsweise Visual Studio .NET nicht mehr unterstützt und somit werden die benötigten Methoden von Visual Studio auch nicht mehr bereitgestellt [MicrFeed].

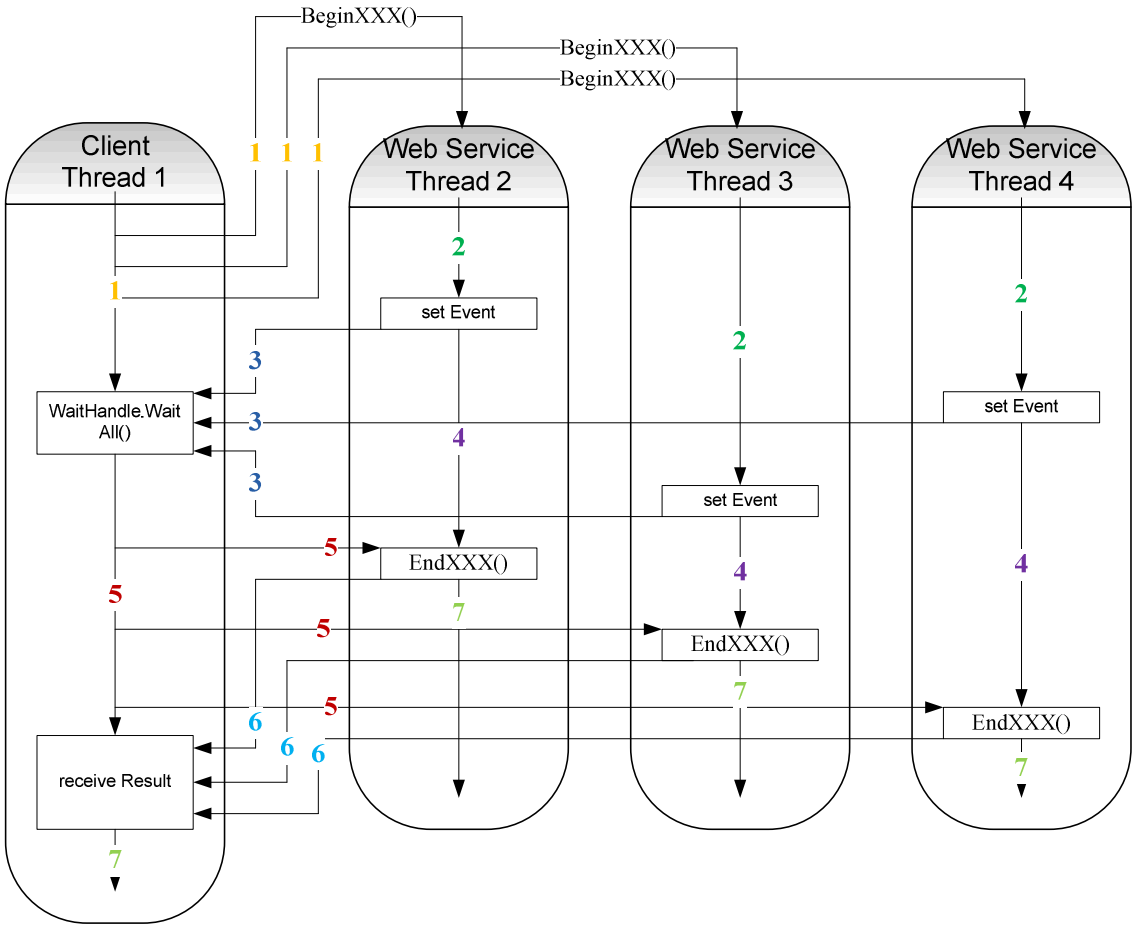

Abbildung 4.3.2-1: Szenario asynchroner Aufruf

Daraufhin wurde entschieden, nur einen Web Service Aufruf mittels des Callback-Szenarios vom Client zu starten. Dies hat den weiteren Vorteil, dass alle Aufrufe, welche Daten aus den Datenbanken abgerufen werden sollen, nun im Web Service implementiert sind. Werden an den Aufrufen Änderungen vorgenommen, muss die COM-Komponente auf dem Client nicht getauscht werden. Des Weiteren wirkt es sich positiv auf die Performance des Systems aus, da nun alle Anfragen an die Datenbank von dem Web Service getätigt werden. Beim ersten Prototyp wurden noch alle Abfragen vom Client aus getätigt. Im Vergleich zu der Geschwindigkeit des ersten Prototyps wurde die Abfragezeit auf im Mittel 4,5 Sekunden gesenkt.

Die Datenübertragung zwischen Web Service und COM-Komponente wurde zunächst in Form von ArrayLists implementiert. Im Laufe der Entwicklung hat sich allerdings gezeigt, dass dieser Typ nicht für die wachsenden Anforderungen des Projekts geeignet ist, da nicht nur Registrywerte sondern auch diverse andere Objekte (siehe 4.4.1:

Datenbanktabelle CompareObject) verglichen werden sollen. Es müssen verschiedene Datentypen transportiert werden und die Menge an zu transportierenden Daten nahm zu. Da die Struktur von ArrayLists komplex ist und nicht auf XML basiert, kann Visual Studio sie nicht automatisch serialisieren. Möchte man sie trotzdem verwenden, muss man die Serialisierung selbst programmieren [Kühn2006]. Wesentlich einfacher ist der Umgang mit DataSets. Sie basieren auf XML und aufgrund dessen wird die Serialisierung automatisch durchgeführt [Kühn2006] [Jone2006] [MSDNCons]. Als Konsequenz wurden die ArrayLists in DataSet-Container verpackt und auf diese Weise übermittelt.
Sicherheitsaspekte wurden bei diesem Prototypen außer Acht gelassen, da das System auf absehbare Zeit nur im Intranet Anwendung finden wird. Einzig die Zugriffskontrolle auf den Web Service sowie die Datenbank ist erhalten geblieben.

4.3.3 Verfehlte Anforderungen

Dieser Prototyp hat die folgenden Anforderungen nicht erfüllt: F4, F10, F11, F12, F13, F14, F16, nFz1, nFz2, nFb14, ... nFb18. Ausschlaggebend war die Verfehlung der Anforderung F4: eine eigene Datenbank für das Projekt wurde unabdingbar, als sich herausstellte, wie umfangreich die Parameter für die Tacho-Vergleiche werden (siehe Datenbank 4.4.1). Hierdurch kam es zur Programmierung eines weiteren Prototyps. Die restlichen Anforderungen, sind zum Teil neu zum anderen Teil alte Anforderungen, welche bis dahin noch nicht umgesetzt worden sind.
Die Anforderungen F4, F11, F12, F13, nFz1, nFz2 ergaben sich in einem Gespräch mit mehreren Kollegen. Nachdem klar war, das ein Vergleich allein auf Registry Werten für die geforderte Funktionalität des Systems (siehe 3.1) nicht ausreichen würde, wurde darüber diskutiert, welche Vergleichsmöglichkeiten berücksichtig werden müssen. Aus diesem anfänglichen Thema entwickelte sich eine Diskussion über das gesamte Projekt. In dieser wurden dann die weiteren Anforderungen deutlich und somit in die Anforderungsliste übernommen.
Die Anforderungen F10, nFb17, nFb18 gehen wieder auf eine Idee des Projektleiters zurück. Er möchte eine statistische Auswertung basierend auf den Informationen des Tacho-Vergleichs. Um diese Auswertung realisieren zu können, wurden die

Anforderungen F10, nFb17 und nFb18 mit in die Anforderungsliste aufgenommen. Die Anforderungen nFb15 und nFb16 ergaben sich in Gesprächen mit Kollegen.

4.4 Der dritte Prototyp – aktueller Stand der Entwicklung

Diese Realisierung basiert auf vier Komponenten: Das Gadget, welches die Schnittstelle zum Benutzer darstellt; einer COM-Komponente, die die Verbindung zwischen Client (Gadget) und Web Service herstellt sowie die Verarbeitung der Daten auf dem Client vornimmt und diese für das Gadget bereitstellt; einem Web Service, der die benötigten Daten auf Anforderung von einem Client aus einer Tacho-Vergleich-Datenbank abruft; und einer Tacho-Vergleich-Datenbank, in der alle benötigten Referenzversionen vorgehalten werden. Die nachfolgende Abbildung stellt die Zusammenhänge anschaulich dar.

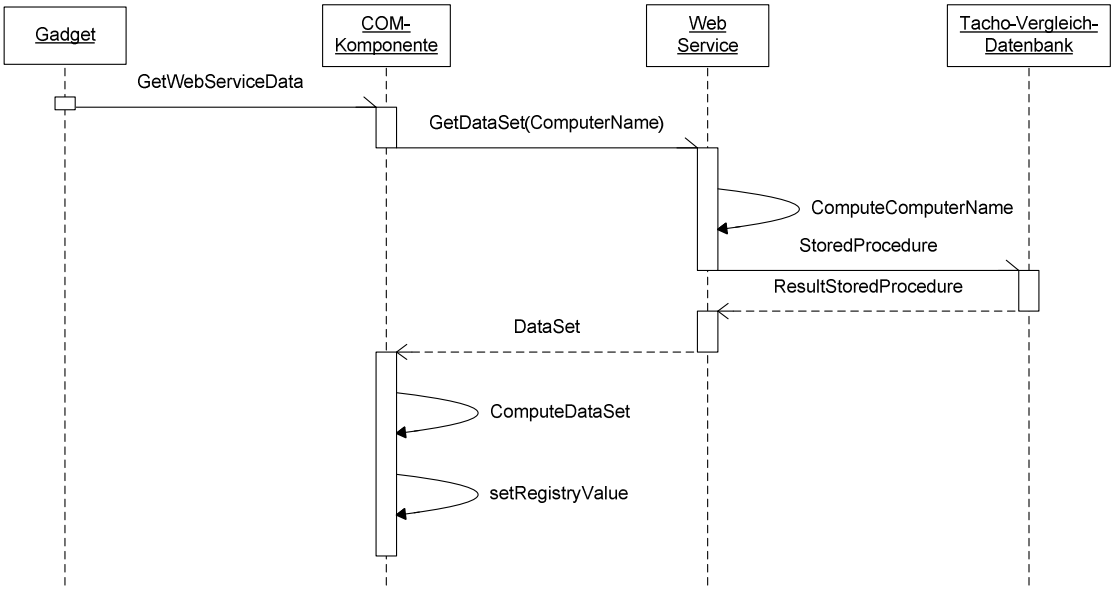

Abbildung 4.4-1: Kommunikation der Komponenten der Realisierung

Folgende Dateien beinhalten den entsprechenden Code:

- Client.cs : Code der COM-Komponente
- WebService.asmx.cs : Code des Web Service
- WSDS.cs : Code des DataSet

Hierbei ist zu beachten, das WSDS.cs eine `partial class` ist. Tatsächlich ist es nur ein Teil einer Klasse. Der andere Teil wird von Visual Studio automatisch angelegt, sobald ein DataSet eingebunden wird. In diesem Teil wir die Verbindung zur Tacho-

Vergleich-Datenbank hergestellt und auch die Ausführung der Stored Procedure ist hier hinterlegt.

4.4.1 Die Tacho-Vergleich-Datenbank

Die Tacho-Vergleich-Datenbank stellt alle Informationen bereit, die für Tacho-Vergleiche, die auf dem Client ausgeführt werden sollen, nötig sind. Sie enthält die Tabellen StandartClient, CompareObject, CompareDescription, CompareWhatToCheck, MetricsFactor and PriorityForDisplay sowie StandartClient. Alle haben eine nummerische ID als Primary Key.

Die Tabelle StandartClient enthält die Spalten Object, Description, WhatToCheck, MetricsFactorTrue, MetricsFactorFalse und PriorityForDisply, die über Foreign Keys auf die entsprechenden anderen Tabellen verweisen. Zusätzlich sind die Spalten Path sowie Value vorhanden, welche den Pfad zum Tacho-Vergleichsobjekt und den Referenzwert für den Tacho-Vergleich aufnehmen. Diese beiden Werte werden von Hand eingetragen. In der Tabelle CompareObject sind die Objekte beschrieben, mit denen der Tacho-Vergleich durchgeführt werden kann. Es sind: File, Directory, RegistryKey, RegistryValue, Process, Service, WMI, localDB (Vergleich von lokaler Datenbank oder Einträgen in ihr) und Program (Programm starten und Returncode auswerten). Implementiert sind in dieser Version auf dem Client der Vergleich auf Dateien, Ordner, Registry Keys und Registry Values. Die Tabelle CompareDescription enthält Informationen über die Art des Tacho-Vergleiches: kleiner, größer, gleich, größer gleich, kleiner gleich, existing, running und IsInstalled. Dies kann zum Teil mit Attributen aus der Tabelle CompareWhatToCheck kombiniert werden: CreationTime, LastWriteTime, LastAccessTime und RunProgram (RunProgram ist auf dem Client noch nicht implementiert). Die Tabelle PriorityForDisplay enthält Prioritäten für die Anzeige auf dem Flyout des Gadgets: 0, 1, 2, …, 9. In der Tabelle MetricsFactor sind Faktoren gespeichert, mit denen ein Tacho-Vergleich gewertet/gewichtet werden kann. Die letzten beiden Tabellen sind in der aktuellen Version auf dem Client noch nicht implementiert.

Die Tacho-Vergleich-Datenbank stellt eine Stored Procedure bereit, um dem Web Service den Zugriff auf die benötigten Daten zu ermöglichen. Die Stored Procedure ist

so konfiguriert, dass nur ein spezieller autorisierter Nutzer diese ausführen darf. Dieser Nutzer ist nur dem Web Service bekannt.

Ein Vorteil dieser Tacho-Vergleich-Datenbank ist es, dass mit dieser Lösung eine Benutzeroberfläche geschaffen werden kann, die es ermöglicht, komfortabel die Daten zu verwalten: Es können Datensätze angelegt, gelöscht und auf Korrektheit überprüft werden, ohne direkt in die Datenbestände eingreifen zu müssen. Durch eine Datenbank, die über Stored Procedures abgerufen werden kann, können Änderungen an ihr vorgenommen werden, ohne dass in die Implementierung des Web Services eingegriffen werden muss, wenn Datensätze hinzugefügt werden müssen. Ein weiterer Vorteil ist, dass alle relevanten Daten in einer Datenbank zusammengeführt werden und nicht über mehrere Datenbanken in zum Teil verschiedenen Tabellen verteilt sind. Folgend eine Abbildung das Tacho-Vergleich-Datenbank Designs (Abbildung 4.4.1-1).

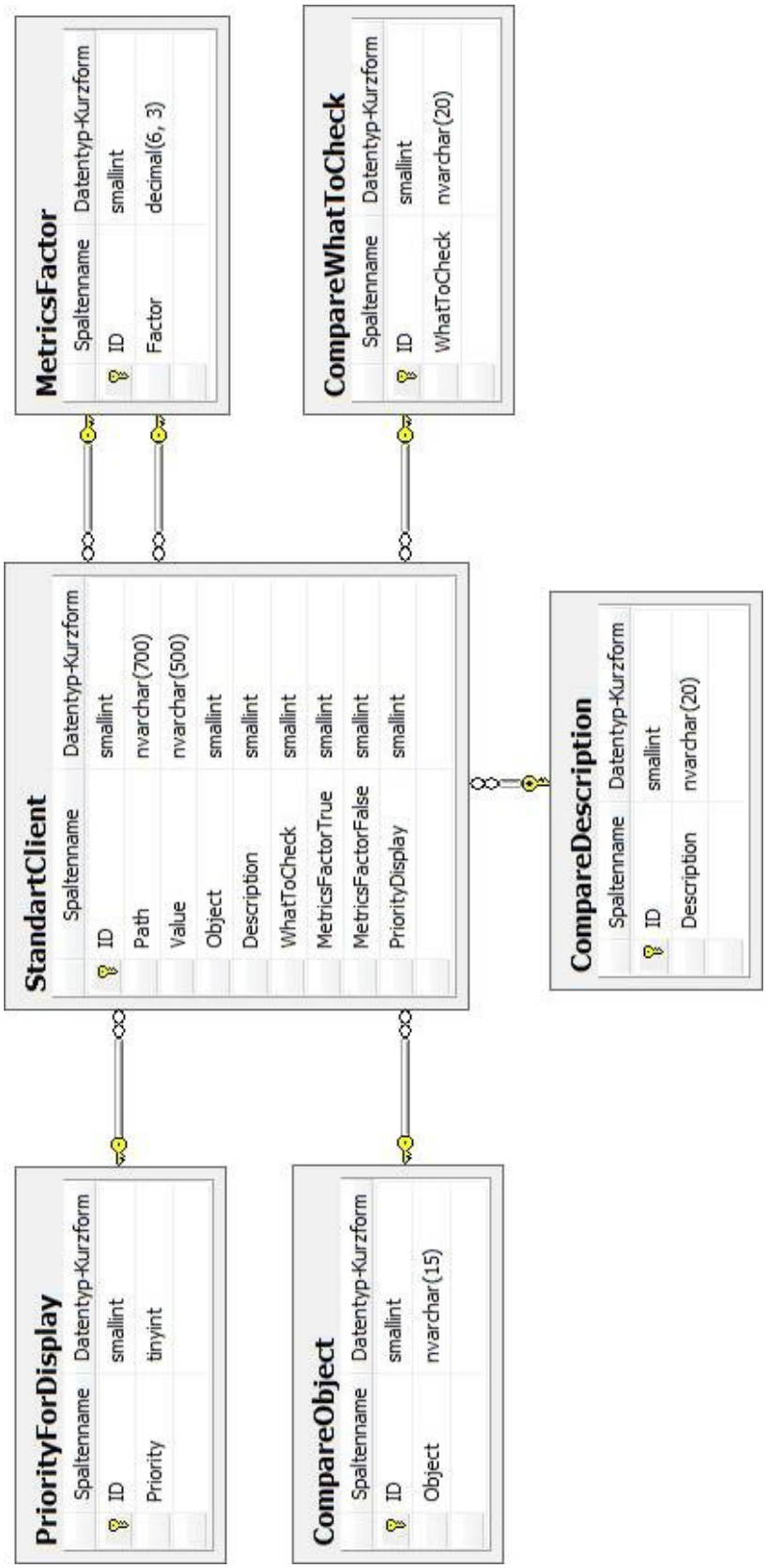

Abbildung 4.4.1-1: Datenbank Design

4.4.2 Der Web Service

Der Web Service wird auf einem IIS 7 (Internet Information Services) bereitgestellt. Der Web Service nimmt Anfragen nur von einem speziellen Nutzer entgegen, welcher nur der COM-Komponente bekannt ist. Als Übertragungsobjekt zwischen Web Service und COM-Komponente wurde das DataSet gewählt. Es lässt sich hervorragend serialisieren und hat die Eigenschaft, komplette Datenbanken (oder Ausschnitte) aufnehmen zu können. Im Vergleich zum zweiten Prototyp wurde die Zwischenspeicherung der Daten aus der Datenbank in einer ArrayList weggelassen. Zum einen genügte die ArrayList, wie zuvor das String-Array, den gestiegenen Anforderungen nicht mehr, zum anderen können die Datensätze auch direkt in einem DataSet gespeichert werden.

Durch eine eigene Datenbank können zusätzliche Informationen integriert werden. Es ist somit nicht nötig, für neue Informationen weitere Stored Procedures vom Web Service ausführen zu lassen. Dies bedeutet, er muss nicht neu implementiert und kompiliert sowie ausgetauscht werden.

Um eine Klasse als Web Service auszuweisen, muss sie das Attribut `[WebService]` aufweisen. In `[WebService]` wiederum muss mit dem Attribut `Namespace` ein eindeutiger Namensraum für den Web Service vergeben werden, damit der Web Service eindeutig identifiziert werden kann. Die Funktionalität des Services wird über Methoden bereitgestellt. Damit diese ansprechbar sind, benötigen sie das Attribut: `[WebMethod]`

```
[WebService(Namespace = "http://www.tuev-nord.de/”]
public class storedProceduresWebServices:
                        System.Web.Services.WebService{

      [WebMethod]
      public DataSet GetDBValues(string computerName){ … }

……
}
```

Der Web Service stellt eine Methode zur Verfügung: “`GetDBValues`“. Diese Methode erwartet als Übergabeparameter einen `computerName` und gibt ein `DataSet` zurück. Anhand des übergebenen Namens wird ermittelt, in was für eine Gruppe der Client einzuordnen ist. Aufgrund dieser Einordung wird von der Tacho-Vergleich-Datenbank

eine spezielle Tabelle, die ausschließlich für die Gruppe von Computern gültig ist, zu der auch dieser Client gehört, abgerufen. Zusätzlich wird noch eine Standardtabelle, die für alle Clients gilt, abgerufen. Der Abruf erfolgt über eine Stored Procedure. Diese ist so konfiguriert, dass nur ein spezieller autorisierter Nutzer sie ausführen darf. Dieser Nutzer ist nur dem Web Service bekannt. Der Abruf erfolgt durch die Methode `FillDS()`. Beide Tabellen werden in einem DataSet verpackt, das von der Methode nach erfolgreicher Ausführung zurückgegeben wird. Der Abruf der beiden Tabellen erfolgt über ein asynchrones Szenario [Kühn2006]. In der aktuell implementierten Version ist nur der Abruf der Standardtabelle möglich.

4.4.3 Die COM-Komponente

Bei der COM-Komponente bestanden die Möglichkeiten, sie als ActiveX-Komponente oder als für COM sichtbare .NET-Komponente [MSDNAct1] zu realisieren. Aufgrund der Einschränkungen von ActiveX unter Windows Vista in Kombination mit der beim TÜV geltenden Group-Policy, habe ich mich für die Entwicklung einer .NET-COM-Komponente entschieden.
Damit die Client Komponente für andere Objekte durch COM instanziierbar ist, müssen die aufrufbaren Funktionen in einem speziellen Interface bekannt gemacht werden. Von diesem Interface wiederum muss die Klasse erben, die die COM Funktionalität bereitstellt. Die Attribute `ComVisible(true)` und `GuidAttribute("XXXXX-XX-XXXXXX-XXX-XXXXXX")` sorgen dafür, dass das Interface und dadurch die Klasse für andere Objekte über COM sichtbar sind. Außerdem werden sie damit unter einem eindeutigen Namen, mit welchem sie auch angesprochen werden können, in der Registry registriert. Für die Klasse ist außerdem das Attribut `[ProgId()]` vorgesehen. Mit diesem ist es möglich, einen natürlich-sprachlichen, lesbaren Namen für die Klasse zu vergeben. Dieser wird auch in der Registry unter der entsprechenden GUID registriert, somit kann die Klasse auch mit diesem Namen angesprochen werden. [Temp2003]

```
[ComVisible(true), GuidAttribute("XXXXX-XX-XXXXXX-XXX-XXXXXX")]
public interface Index {
      int GetIndex();
      string GetIndexText();
      void RunWebServices();
```

```
        int GetFlag();
    }

    [ComVisible(true), GuidAttribute("XXXXX-XX-XXXXXX-XXX-XXXXXX")]
    [ProgId("TNG.gadget.lokalerComputerstatus")]
    public class GadgetInterface : Index{ … }
```

Jetzt ist es möglich, nach erfolgreicher Instanziierung, die angebotenen Funktionen zu nutzen. Dieses Interface stellt Funktionen zum Abruf des Ergebnisses des Web Services, zum Aufruf des Web Services sowie zur Abfrage eines Registrywertes bereit.
Der Aufruf des Web Services über `RunWebServices()` erfolgt asynchron durch ein Callback-Szenario [Morg2007]. Dabei ist `objWSDBData` ein Objekt des WebServices, an dessen Attribut `GetDBValuesCompleted` ein Event geheftet wird. Durch dieses Event wird die Methode `computeDS` des Objektes `objComputeWebService` ausgeführt, sobald der asynchrone Aufruf des Web Services zurückgekehrt ist. Der Aufruf selbst erfolgt durch `objWSDBData.GetDBValuesAsync`, wobei als Parameter noch der Computer Name mit übergeben wird: `objComputeWebService.GetComputerName()`. Die Methode `GetDBValuesAsync()` und auch der EventHandler `GetDBValuesCompletedEventHandler` werden von Visual Studio automatisch erzeugt, wenn eine Referenz auf einen Web Service angelegt wird. Sie sorgt für einen asynchronen Aufruf, ohne dass sich der Programmierer mit der dahinterliegenden Thematik beschäftigen muss.

```
objWSDBData.GetDBValuesCompleted += new
    Client.WebService_localhost.GetDBValuesCompletedEventHandler(
    objComputeWebService.computeDS);

objWSDBData.GetDBValuesAsync(objComputeWebService.
    GetComputerName());
```

Beim Erzeugen des Objektes `objComputeWebService` wird der Konstruktor dieser Klasse aufgerufen. Dieser erzeugt ein DataSet `resultDS`, welches die Daten der Ergebnisse aufnehmen soll, um sie später wieder zurück in eine Tacho-Vergleich-Datenbank schreiben zu können.
Die Methode für die Verarbeitung des Web Services `computeDS` nimmt das Ergebnis des Web Services in einem DataSet entgegen. In diesem DataSet befindet sich nun ein Client spezifischer Ausschnitt der Tacho-Vergleich-Datenbank. In diesem wiederum ist

eine Tabelle hinterlegt, die die für den Tacho-Vergleich benötigten Daten enthält. Mittels einer for-Schleife wird die Tabelle zeilenweise iteriert. Für jede Zeile wird die Methode (`BeginCompare()`) asynchron aufgerufen, die den Vergleich der Referenzdaten aus der Tacho-Vergleich-Datenbank mit den Daten des Clients durchführt. Der Aufruf erfolgt wieder nach dem Szenario aus [Kühn2006].
In diesem Fall muss allerdings kein Ergebnis nach dem asynchronen Aufruf der Methode abgerufen werden. Alle asynchron aufgerufenen Methoden schreiben ihr Ergebnis in das oben erwähnte DataSet `resultDS`, welches `static` ist. Aus diesem Grunde wird die Methode, mit der auf das Ergebnis nach der asynchronen Ausführung zugegriffen wird, nicht benötigt. Ebenfalls abweichend ist der Einsatz von `ManualResetEvent[]`. Hiermit wir manuell ein Event erzeugt. Dieses Event zeigt an, dass die Methode mit ihrem Vergleich fertig ist. Alle Events dieses Typs werden in `manualResetEvents[]` gesammelt, um mit `WaitHandle.WaitAll(manualResetEvents[],…)` auf die Beendigung aller asynchron aufgerufenen Methoden zu warten. Darauf folgend beginnt die Auswertung der Ergebnisse, um den Tachowert in die Registry zu schreiben und eine textuelle Darstellung der Ergebnisse zu erzeugen. Dieser Text wird auf dem Flyout des Gadgets dargestellt, um dem TÜV-Nutzer Maßnahmen aufzuzeigen, die er ergreifen kann, um erkannte Probleme zu lösen. Um diesen Text abzurufen, wird vom Gadget die Funktion `GetIndexText()` aufgerufen. Um den Tachowert auszulesen, ruft das Gadget `GetIndex()` auf. Unter Punkt 3.1 wird eine Funktion beschrieben, mithilfe derer sich die Oberfläche des Gadgets orange einfärben lässt. Dies beruht auf einem Registry Wert. Um diesen auszulesen, ruft das Gadget die Funktion `GetFlag()` auf.

4.4.4 Der UDDI-Service

Der UDDI-Service stellt keine Funktionalität bereit, die für diese Projekt von Nutzen ist. Da die Informationen, welche vom Gadget benötigt werden, sehr speziell sind, ist nicht davon auszugehen, dass sie in einem anderen Projekt Verwendung finden werden. Aufgrund dessen muss der Web Service nicht bei einem UDDI-Service registriert sein.
Um die Routing-Funktionalität dennoch zur Verfügung zu haben, bietet sich die Implementierung eines WSE-Routers an. Er bietet weiterhin die Vorteile,

Sicherheitsaspekte wie verschlüsselte Übertragungen und Authentifizierung mittels Zertifikaten zu unterstützen sowie Ausfallzeiten des Web Services zu minimieren.

5 Fazit und Ausblick

5.1 Ausblick

Dieses Kapitel behandelt die Validierung, die möglichen Erweiterungen aufgrund der noch nicht erfüllten Anforderungen und einen Ausblick auf eine neue .NET Kommunikationstechnologie.

5.1.1 Validierung

Die Validierung bezeichnet den Prozess des Überprüfens und Analysierens. Dieser stellt sicher, dass eine Software mit ihrer Spezifikation übereinstimmt und die Bedürfnisse des Kunden erfüllt. Hierzu gehören auch Anforderungs-Reviews, die in freier Form während der Entwicklung der Prototypen stattgefunden haben (siehe 4.2.3 und 4.3.3).
Um systematische Tests durchzuführen, müssen laut [Somm2001] die folgenden Arten von Testverfahren angewandt werden:

- Black-Box-Tests: Bei diesem Verfahren werden die Tests anhand von Spezifikationen durchgeführt. Der Tester hat keinen Zugriff auf den zugrunde liegenden Code. Er nimmt Eingaben vor und untersucht die entsprechenden Ausgaben. Entspricht die Ausgabe nicht der Vorhersage, so ist der Test erfolgreich verlaufen, denn er hat einen Fehler aufgedeckt.
- Strukturelle Test (White-Box-Tests): Dies ist ein Testansatz, bei dem der Code dem Tester vorliegt. Er kann den Code analysieren und auf dieser Grundlage Testdaten erstellen. Diese Art von Test ist allerdings bei eher kleinen Programmeinheiten üblich, beispielsweise einzelne Methoden einer Klasse.

Die genannten Verfahren können eingesetzt werde, um laut [Somm2001] folgende Ziele zu realisieren:

- Modultests, da das System aus verschiedenen Komponenten aufgebaut ist. Mit Visual Studio ist dies für .NET mit dem Unit Testing Framework [MSDNUnit] möglich. Hiermit können Tests geschrieben und dann automatisiert ausgeführt

werden. Hierfür eignen sich nach Meinung des Autors vor allem Fehlertests und Pfadüberdeckungstests.

- Integrationstests, da das System auf Basis verschiedener Komponenten, die abhängig voneinander sind, realisiert worden ist. Um das Zusammenwirken der einzelnen Komponenten zu testen, muss das System als Ganzes getestet werden. Da mit der Anzahl der Komponenten auch der Aufwand diese zu testen steigt, kann man auch spezielle Szenarien für einzelne Subsysteme testen: Im Falle dieser Arbeit zum Beispiel den Abruf der Daten aus der Datenbank durch den Web Service, die Anzeige des Vergleich-Tacho Wertes auf dem Gadget und damit verbunden den Funktionsaufruf der COM-Komponente oder den Aufruf des Web Services durch die COM-Komponente.
- Systemtests, um losgelöst von der Komponentensicht das gesamte System beurteilen zu können. Hierzu zählt auch die Benutzbarkeit.
- Belastungstest, insbesondere bei der Ausführung des Abrufes der Daten aus der Tacho-Vergleich-Datenbank. Wie verhält sich das System, wenn die Datenbank extrem viele Anfragen zu handeln hat, ebenso der Web Service? Ist die Funktion unter diesen Bedingungen noch gewährleistet?

Anhand der genannten Tests wird deutlich, dass in der Planung für ein Projekt auch ein entsprechend großer Zeitraum für die Entwicklung und Durchführung der Test bedacht werden sollte. Nur durch eine gründliche und vollständige Validierung ist sichergestellt, dass das ausgelieferte Produkt den Vorstellungen des Kunden entspricht und keine vermeidbaren Fehler mehr enthält.

Aufgrund des eingeschränkten zeitlichen Rahmens dieser Arbeit war es nicht möglich, eine erforderliche vollständige Validierung durchzuführen. Es wurden bisher Tests am Gadget bezüglich der Anforderungen nFb4, nFb5, nFb8, … , nFb12 durchgeführt und dadurch bestätigt. Die durchgeführten Tests können am ehesten als Fehlertests bezeichnet werden [Somm2001]. Es wurden ebenfalls Messungen bezüglich der Performance (Anforderung nFe1) im Bereich der Datenbereitstellung über das hausinterne Netzwerk vorgenommen. Dies wurde durchgeführt, indem die Zeit vom Verbindungsaufbau der COM-Komponente mit dem Web Service bis zum Eintreffen der Antwort des Web Services bei der COM-Komponente gemessen wurde. Zur Durchführung der Messung wurde eine .NET Klasse benutzt [MSDNStop]. Dieser Test

bewirkte die Entwicklung eines zweiten Prototyps, da der erste Prototyp die relevante Anforderung nicht erfüllte. Die Ausführung dieser Tests hat insgesamt ungefähr anderthalb Tage beansprucht.

5.1.2 Erweiterungsmöglichkeiten

Da auch der dritte Entwurf bisher noch ein Prototyp ist und noch nicht alle Anforderungen umgesetzt hat, bleiben noch diverse Möglichkeiten für Erweiterungen. Die graphische Darstellung ist in vielen Bereichen noch nicht hinreichend realisiert (Anforderungen nFb13, nFb14, nFb15, nFb17, nFb18). Im Bereich des Flyouts auf dem Gadget könnte dies durch eine Art Tab-Zeile erreicht werden, um zwischen verschiedenen Informationen/Ansichten umschalten zu können. Für die Konfiguration der Datensätze in der Tacho-Vergleich-Datenbank wird noch eine graphische Oberfläche benötigt. Diese wird dann auch eine Überprüfung der eingegebenen Daten durchführen. In der Tacho-Vergleichs-Datenbank werden noch weitere Tabellen benötigt, die für die unterschiedlichen Gruppen von Clients gelten (F11). Es wird eine weitere Datenbank benötigt, welche die Ergebnisse der Vergleiche aufnimmt (F10). Es müssen noch Regeln für eine Gewichtung der Vergleiche aufgestellt werden (F9). Es ist noch kein Verfahren entwickelt worden, um die zu übertragende Datenmenge zwischen Client und Web Service zu reduzieren (F12). Des Weiteren muss die Behandlung von Fehlermeldungen des Systems korrekt implementiert werden (F13). Es muss die Möglichkeit der Programmausführung auf dem Client geschaffen werden (F14) und es muss die Authentifizierung der Clients gegenüber dem Web Service durch Computerzertifikate eingeführt werden (F16). Des Weiteren fehlt noch eine Funktion, mit der es möglich ist, Änderungen an den Datensätzen der Tacho-Vergleich-Datenbank nur einem ausgewählten Kreis von TÜV-Nutzern zur Verfügung zu stellen, um eventuelle Fehler erkennen zu können, bevor sie im ganzen Unternehmen verteilt werden (nFz1). Die Erreichbarkeit des Web Services ist auch noch nicht sichergestellt (nFz2), dies kann durch die Realisierung eines WSE-Routers geschehen. Sobald das Service Pack 1 für Windows Vista erscheint, muss die Zuverlässigkeitsüberwachung in den Tacho-Vergleich mit einbezogen werden.

5.1.3 WCF – ein neues .NET Kommunikationsmodell

Mit Erscheinen von .NET 3.0 hat Microsoft die WCF (Windows Communication Foundation) eingeführt. Mit WCF ist ein standardisiertes Programmiermodell verfügbar, unter dem alle Vorteile von Kommunikationstechnologien der Vorgänger (zum Beispiel Distributed COM, Enterprise Services oder Web Services) vereint und vereinheitlicht sein sollen. Hinzu kommt, dass hierdurch eine Kommunikation mit anderen Plattformen gewährleistet ist.

Die WCF will gewährleisten, dass sich die Entwickler nicht mehr um die Implementierung des Datenaustausches kümmern müssen, sondern diesen frei konfigurieren können, um sich ganz auf die Programmierung der Logik des Programms zu konzentrieren. Die WCF stellt dabei die Laufzeitumgebung für die Programmlogik bereit, während die Kommunikation eigenständig von der WCF durchgeführt wird und vom Entwickler nach seinen Bedürfnissen konfiguriert werden kann. Die Kommunikation von verteilten Anwendungen wird dabei auf der Basis einer Service orientierten Architektur durchgeführt.

Die Migration von bestehenden .NET-Web Service Anwendungen nach WCF soll problemlos durchzuführen sein, laut Aussage des Autors, da beide serviceorientierte Technologien darstellen.

Weiterhin gibt es diverse neue Features, die man vor WCF von Hand programmieren musste. Auch die Performance soll sich, laut Aussage des Autors, deutlich erhöht haben. [Kotz2007]

5.2 Erfahrungen

In diesem Kapitel werden die Erfahrungen, die im Laufe des Projektes gesammelt wurden, geschildert. Sie können hilfreich sein bei der Entwicklung von Software nach einer prototypischen Vorgehensweise sowie in einem Projekt, das die Implementierung von Web Services unter .NET 2.0 verfolgt.

5.2.1 Bewertung von .NET Web Services

Die Implementierung von .NET Web Services mit Visual Studio 2008 ist sehr komfortabel. Es wird die gesamte Ebene des SOAP für den Entwickler verdeckt. Dies funktioniert sehr gut, da Visual Studio alle benötigten Funktionen zuverlässig bereitstellt. Wenn man .NET Web Services einsetzten möchte, sollte man sich vorher Gedanken darüber machen, welche Datentypen zu übertragen sind. Eine Vielzahl von Datentypen wird von .NET automatisch serialisiert [MSDNData], bei allen Anderen muss man hingegen die Serialisierung von Hand implementieren [MSDNGrun].
NET Web Services haben sich im Rahmen dieses Projektes bewährt. In Kombination mit einer .NET COM-Komponente auf dem Client wurde die gesamte Kommunikation vor dem Entwickler verdeckt und hat trotzdem hervorragend funktioniert. Das Bereitstellen von .NET Web Services auf einem IIS ist, sofern der IIS erst einmal korrekt eingerichtet ist, sehr einfach.

5.2.2 Bewertung der prototypischen Vorgehensweise

Eine Bewertung der prototypischen Vorgehensweise findet auf Basis dieser Arbeit statt und kann nicht mit anderen Vorgehensweisen verglichen werden, da von Seiten des Autors keinerlei praktische Erfahrung in anderen Vorgehensweisen besteht. Die Entwicklung von Prototypen kann nur dann erfolgen, wenn genügend Zeit für die Entwicklung zur Verfügung steht. Es muss eingeplant werden, komplette Prototypen zu verwerfen und mit der Programmierung weiterer ganz neu zu beginnen. Diese Zeit steht bei einem festen Abgabetermin unter Umständen nicht zur Verfügung. Des Weiteren kostet eine längere Entwicklungszeit in den meisten Fällen auch mehr Geld. Auch dieser Faktor muss in die Planung mit einfließen. Da das Resultat dieser Arbeit

innerhalb des Konzerns eingesetzt wird und keine zeitliche Befristung zur Implementierung gegeben ist, entstehen durch den längeren Entwicklungszeitraum keine unvorhergesehenen höheren Kosten.

Auch wenn nur wenige konkrete Anforderungen für einen ersten Prototypen zur Verfügung stehen, muss man versuchen, diese mit dem Kunden so genau wie möglich zu formulieren. Ansonsten kann es passieren, dass man einen hervorragenden Prototyp entwickelt hat, dieser aber in gewissen Punkten gar nicht den Vorstellungen des Kunden entspricht.

Bevor man mit Programmieren beginnt, sollte man sich auf jeden Fall einen groben Entwurf theoretisch entwickelt haben. Ohne diesen Entwurf kann es passieren, dass man den Überblick verliert und in Sackgassen gerät, aus denen man nicht mehr herauskommt. Dies kann durch einen Entwurf fast immer verhindert werden.

Bei der Entwicklung weiterer Prototypen muss man überdenken, was von dem alten Prototyp übernommen werden kann. Manchmal ist es sinnvoller, schon erstellte Teile neu zu programmieren als sie anzupassen. Durch das Anpassen wird der Code häufig unleserlich und damit schlecht wartbar. Falls der Code dann in einem weiteren neuen Prototyp zum Einsatz kommen soll, wird die Situation wahrscheinlich noch verschlimmert.

Die Entwicklung der Prototypen hat entscheidend dazu beigetragen, dem Kunden sowie dem Autor eine detailliertere und vollständigere Sicht auf die Anforderungen des zu implementierenden Systems zu verschaffen. Dies lässt sich anhand der Vervollständigung der Anforderungen nachvollziehen (siehe 3.2.2). Durch die Entwicklung der Prototypen sind verschiedene Realisierungsmöglichkeiten aufgetan worden (siehe 3.3, 4.2, 4.3, 4.4). Diese konnten dadurch in der Praxis auf ihre Eignung bezüglich der Anforderungen und somit des Projektes getestet werden. Die prototypische Vorgehensweise ergibt nur Sinn, wenn regelmäßig über die aktuellen Entwicklungen gesprochen wird (Review, siehe 4.2.3, 4.3.3), mit dem Kunden sowie, im Falle dieser Arbeit, mit den Kollegen. Dabei ist es allerdings nicht ausreichend, den aktuellen Prototyp vorzuführen und auf technische Details hinzuweisen. Man muss immer das gesamte zu entwickelnde System betrachten und darauf gestützt die Gespräche führen. Hierbei muss vor allem der Kunde im Mittelpunkt stehen und sich äußern können

In einer Firma, in der man sich als Neuzugang erst noch in den Kollegenkreis integrieren muss, können diese Gespräche über Fachthemen auch dazu genutzt werden, neue Kontakte zu knüpfen und dadurch einander besser kennenzulernen.
Im Falle dieser Arbeit waren die Anforderungen zu Beginn nur grob spezifiziert. Dadurch ist man als Programmierer in der Lage, selbst am Gestaltungsprozess des Systems und somit der Anforderungen aktiv teilzunehmen. Dies ist sicherlich interessanter, als ein System nach fest vorgegebenen Anforderungen zu entwickeln.

5.2.3 Allgemeine Probleme während der Arbeit

Im Verlauf der Vorbereitung und auch der Anfertigung dieser Arbeit gab es eine Reihe von Problemen zu bewältigen. Einige haben nicht lange aufgehalten, andere waren extrem hartnäckig. Dennoch haben sie alle zum Lernprozess beigetragen. Einige sehr markante Beispiele seien hier namentlich genannt:
Möchte man einen Web Service asynchron aufrufen, so wird man quasi im ganzen Internet, auf allen MSDN (Microsoft Developer Network)- und Technet-Seiten sowie in einem aktuellen Trainingsbuch von Microsoft [MicrFeed], mit einem Beispiel konfrontiert, das nicht mehr unterstützt wird. In all diesen Quellen wird aber explizit darauf hingewiesen, dass eine Unterstützung, auch mit den entsprechenden Werkzeugen, gewährleistet sei. Man könnte dadurch an sich und seiner Implementierung zu Zweifeln beginnen. Nach eingehender Recherche, ungefähr einen Tag lang, wurde dann die Lösung gefunden, weit versteckt in den Tiefen des Internets: [MicrFeed].
Es wurde eine virtuelle Maschine benötigt, um einige Dinge mit WSE ausprobieren zu können. Leider wird WSE von Visual Studio 2008 nicht mehr unterstützt, sodass Visual Studio 2005 benötigt wurde. Da Probleme vorauszusehen sind, wenn VS 2005 nach VS 2008 installiert wird, musste ein zweites Betriebssystem in einer virtuellen Maschine eingerichtet werden. Dieses Problem hat ungefähr einen halben Tag Arbeitszeit gekostet.
.NET Web Services sind in der Framework Version 2.0 nur unter einem IIS lauffähig. Zu Testzwecken wurde ein IIS 7 lokal auf der Entwicklungsmaschine eingerichtet und ein weiterer IIS 6 auf einem anderen PC. Die Konfigurationsmöglichkeiten eines IIS

sind ausgesprochen vielfältig. Es vergingen fast zwei Tage, bis beide Server so konfiguriert waren, dass der Web Service auf beiden voll funktionstüchtig lief.

5.3 Fazit

5.3.1 Bewertung des Einsatzes der Lösung im Unternehmen

Ein System, das pro-aktiv dafür sorgt, Arbeitszeitverluste zu minimieren, ist sicherlich eine wichtige Komponente in einem Unternehmen. Wenn dieses System vollautomatisch dafür sorgt, mögliche Probleme zu erkennen, darauf aufmerksam zu machen sowie entsprechende Gegenmaßnahmen einzuleiten und somit Geld einzusparen, wird es schon nach kurzer Zeit eine wichtige Stellung im Unternehmen eingenommen haben. Es wird ebenfalls die Arbeit der Administratoren erleichtern. Sie können jetzt vorbeugend handeln, was weniger Zeit in Anspruch nehmen wird, als die Lösungsprozesse eingetretener Probleme, und sich somit auf andere Tätigkeiten konzentrieren.

Das System wurde so konzipiert, dass es leicht bedienbar ist. Dadurch wird nur minimaler Schulungsaufwand für die TÜV-Nutzer nötig sein. Dies trägt zur Akzeptanz des Systems unter den TÜV-Nutzern bei. Diese Akzeptanz ist wichtig, da das System unter anderem auf die Mitarbeit der TÜV-Nutzer angewiesen ist. In diesem Zusammenhang ist es ebenfalls wichtig, den TÜV-Nutzern glaubhaft versichern zu können, dass nur der Computer überwacht wird und nicht sie selbst. Hierzu wird der Betriebsrat beitragen, denn bei jeglicher Art von Überwachungssoftware ist er mitbestimmungspflichtig.

Literaturverzeichnis

Printmedien:

[Dust2003] Dustdar, Schahram; Gall, Harald; Hauswirth, Manfred; Software Architekturen für verteilte Systeme; Springer; 2003; ISBN: 1439-5428

[Eber2003] Eberhart, Andreas; Fischer, Stefan; Web Services – Grundlagen und praktische Umsetzung mit J2EE und .NET; Hanser; 2003; ISBN: 3-446-22530-7

[Hamm2005] Hammerschall, Ulrike; Verteilte Systeme und Anwendungen, Architekturkonzepte, Standarts und Middleware-Technologien; Pearson Studium; 2005; ISBN: 3-8273-7096-5

[Jone2006] Jones, Allen: Microsoft Visual C# .NET Programmier-Rezepte; Microsoft Press; 2006; ISBN: 3-86063-091-1

[Kotz2007] Kotz, Jürgen; Haban, Rouven; Steckermeier, Simon: .NET 3.0 – WCF, WPF und WF – ein Überblick; Addison-Wesley; 2007; ISBN: 978-3-8273-2493-1

[Kühn2006] Kühnel, Andreas : Visual C# 2005 Das umfassende Handbuch; Galileo Computing; 3., aktualisierte und erweiterte Auflage 2006; ISBN: 978-3-89842-586-5

[Lieb2007] Liebhart, Daniel; SOA goes real, Service-Orientierte Architekturen erfolgreich planen und einführen; Hanser; 2007; ISBN: 3-446-41088-0

[Morg2007] Morgan, Sara; Ryan, Bill; Horn, Shannon; Blomsma, Mark: Entwickeln von verteilten Anwendungen mit .NET Framework 2.0; Microsoft Press; 2007; ISBN: 3-86645-912-2

[Schi2007] Schill, Alexander; Thomas Springer: Verteilte Systeme; Springer; 2007, ISBN: 1614-5216

[Somm2001] Sommerville, Ian; Software Engineering; Pearson Studium; 2001; 6. Auflage; ISBN: 3-8273-7001-9

[Temp2003] Templeman, Julian; Mueller, John Paul: COM Programming with Microsoft .NET; Microsoft Press; 2003; ISBN:0735618755

Hyperlinks:

[ISO20000] ISO 20000: Abruf: 25.02.2008

[ISOIEC91] ISO/IEC TR 9126-4:2004, Abruf: 08.03.2008

[itSMFITL] itSMF: Was ist ITIL?, Abruf: 19.02.2008

[mbufMicr] mbuf – Microsoft Business User Forum e.V., Abruf: 19.02.2008

[MicrActX] Microsoft: Description of ActiveX Technologies, Abruf: 16.02.2008

[MicrCOM] Microsoft: Component Object Model Technologies, Abruf: 28.02.2008

[MicrFeed] Microsoft: Begin/End Async WebService Proxy Methods Not Generated in Web Application Projects, Abruf 04.03.2008

[MicrKun1] Microsoft: Kundenreferenzen, TÜV NORD Helpdesk Festplattenfehler, Abruf: 20.02.2008

[MicrTAP] Microsoft: Technical Adoption Program (TAP), Abruf: 19.02.2008

[MSDNAct1] MSDN: ActiveX- und COM-Komponenten, Abruf: 02.02.2008

[MSDNAct2] MSDN: Powering Your Gadgets With ActiveX, Abruf: 16.01.2008

[MSDNCons] MSDN: Consuming a DataSet from an XML Web Service, Abruf: 01.02.2008

[MSDNData] MSDN: Data Types Supported by XML Web Services Created Using ASP.NET, Abruf: 12.03.2008

[MSDNDot1] MSDN: .NET Framework Class Library, Abruf:28.02.2008

[MSDNGadg] MSDN: Windows Sidebar Object Reference, Abruf: 15.02.2008

[MSDNGrun] MSDN: Grundlagen der Serialisierung in .NET Framework, Abruf: 12.03.2008

[MSDNOut1] MSDN: Microsoft Office Outlook 2003 Visual Basic Reference, Abruf: 03.03.2008

[MSDNStop] MSDN: StopWatch Class, Abruf: 10.03.2008

[MSDNUnit] MSDN: Unit Testing Framework, Abruf: 10.03.2008

[OASISUDD] OASIS: UDDI Specification, Abruf: 14.02.2008

[OASISUDI] OASIS: Introduction to UDDI - Important Features and Functional Concepts, Abruf: 04.03.2008

[OASISWe1]	OASIS: Web Services Security, Abruf: 14.02.2008
[RFC 2045]	RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, Abruf: 09.02.2008
[RFC 4648]	RFC 4648: Base64-Kodierung, Abruf: 09.02.2008
[SAPRampU]	SAP Ramp-Up, Abruf: 19.02.2008
[TECHDerA]	TechNet: Der ActiveX-Installationsdienst in Windows Vista, Abruf: 02.02.2008
[TechGadg]	TechNet: Skripting für Windows Vista - Erstellen von Gadgets, Abruf: 28.02.2008
[TÜVNORDW]	Wir über uns: Die TÜV NORD Gruppe, Abruf: 31.01.2008
[W3CSOAP]	W3C: Latest SOAP versions, Abruf: 14.02.2008
[W3CWebS1]	W3C: Web Services Architecture Requirements, Abruf: 09.02.2008
[W3CWebS2]	W3C: Web Services Activity, Abruf: 14.02.2008
[W3CWebS3]	W3C: Web Services Addressing, Abruf: 14.02.2008
[W3CWebS4]	W3C: Web Services Description Language, Abruf: 14.02.2008
[WikipAct]	Wikipedia: ActiveX, Abruf: 28.02.2008
[WikipCOM]	Wikipedia: Component Object Model, Abruf: 28.02.2008
[WikipIS1]	Wikipedia: ISO/IEC 9126, Abruf: 08.03.2008
[WikipITI]	Wikipedia: IT Infrastructure Library, Abruf: 19.02.2008
[WikipNET]	Wikipedia: .NET, Abruf: 28.02.2008

Glossar

Active Directory	Microsoft: Verzeichnisdienst
Assembly	Microsoft: Übersetztes und somit ausführbares .NET Programm
Base64-Kodierung	Dient zum Übertragen von 8-Bitgruppen über nicht 8-bitsichere Wege. Es werden jeweils drei Bytes zu einer 24-Bitfolge zusammengefasst und diese dann in vier 6-Bitgruppen unterteilt. Diese werden an Hand einer Zeichentabelle (Base64-Alphabet) zurückübersetzt und übermittelt. Dadurch wird jeder Text um 33% länger[RFC 2045] [RFC 4648]
COM-Komponente	Microsoft: Ermöglicht Interprozesskommunikation und dynamische Objekterzeugung
DataSet	Microsoft, .NET: im Hauptspeicher verwaltete zusammengehörige Datenmenge in tabellarischer Struktur
DLL	Microsoft: Bezeichnet allgemein eine dynamische Bibliothek
EventLog	Microsoft: Einträge in der Ereignisanzeige von Windows Systemen
Flyout	Microsoft: Popup-Fenster, welches zusätzliche Informationen darstellen kann. Teil eines Gadgets
Gadget	Microsoft: Kleines Programm(JavaScript, HTML), darstellbar auf der Sidebar
Garbage Collection	Automatische Speicherbereinigung; nicht mehr benötigter Speicher wird automatisch erkannt und freigegeben.

Group-Policy	Microsoft: Group-Policy's sind Sammlungen von Benutzer- und Computerkonfigurationseinstellungen, die mit Computern, Standorten, Domänen oder Organizational Units verknüpft werden können, um das Verhalten des Benutzerdesktops zu steuern, Dinge wie Sicherheitseinstellungen, Anmelde- und Abmeldeskripte, Skripte für den Start und das Herunterfahren eines Computers zu definieren oder z.B. Ordnerumleitungen festzulegen. Man kann das Verhalten des Betriebssystems bestimmen und dessen Optionen einschränken. Es gibt auch Group-Policy's, mit denen das Verhalten und die Optionen von Anwendungen von zentraler Stelle aus gesteuert werden können.
HP OpenView	Hewlett Packard: Softwareportfolio zum Verwalten und Überwachen der IT-Infrastruktur großer Unternehmen
HTTP POST	Übertragung von Daten durch HTTP POST. Daten befinden sich nicht in der URL, deshalb können auch große Datenmengen übertragen werden
HTTP Response	Antwort eines Servers auf eine Client-Anfrage. Besteht aus Statuscode mit erläuterndem Text und Header-Informationen. Diese enthalten Angaben über den Server sowie Meta-Daten über das angeforderte Objekt. Danach folgen die eigentlichen Daten
IIS	Microsoft: Bezeichnung für bestimmte Funktionen zur Veröffentlichung von Dokumenten und Dateien im Internet und Intranet
Just-In-Time (JIT) Compiler	Der JIT-Compiler übersetzt während der Laufzeit bei Bedarf den Zwischencode in einen nativen Maschinen-Code

Namespace	Ein Name identifiziert ein Objekt. Für eine eindeutige Zuordnung ist jedoch der entsprechende Kontext, der Namensraum (oder Namespace), zu beachten
Organizational Unit	Microsoft: ein Containerobjekt, das zum Gruppieren anderer Objekte im Active Directory dient
Outlook: Anzeigen als	Microsoft: Wird im Kalender mit der auswählten Farbe angezeigt. frei: weiss; blau-weiss: mit Vorbehalt; blau: beschäftigt; rot: abwesend
Registry	Microsoft: Zentrale, hierarchische Konfigurationsdatenbank von Windows Systemen
RPC	Mit Hilfe von RPC können über ein Netzwerk Funktionsaufrufe auf entfernten Rechnern durchgeführt werden.
SCOM	Microsoft: zentralisierte Verwaltung von Hard- und Software innerhalb eines Unternehmens
Service Desk	Service Desk ist zentrale Anlaufstelle für alle IT-Angelegenheiten einer Institution. Er stellt Funktionseinheit im Rahmen des IT Service Managements dar und garantiert Erreichbarkeit der IT-Organisation für ihre Nutzer.
Sidebar	Microsoft: Befindet sich an der Seite des Desktops von Windows Vista. Stellt die Anzeige von Gadgets zur Verfügung.
UDDI	OASIS: ein standardisierter Registrierungsdienst für Web Services
Unit Test	Er dient zur Verifikation der Korrektheit von Modulen einer Software, z.□B. von einzelnen Klassen
Windows Script Host	Microsoft: eine COM-basierte Laufzeitumgebung für Skriptsprachen

WMI	Microsoft: Über WMI kann lesend und schreibend, lokal oder entfernt, auf nahezu alle Einstellungen eines Windows-Computers zugegriffen werden.
WSE	Microsoft: Add-On für Visual Studio und .NET Framework. Updates für neue Web Service Spezifikationen
WSE-Router	Microsoft: ein Web Service der Routing- und Sicherheitsfunktionen als Vermittlungsstelle zwischen Web Services und Clients bereitstellt
Zuverlässigkeitsüberwachung	Microsoft: registriert alle Systemfehler unter Windows Vista und ermöglich Zugriff auf diese

Abkürzungsverzeichnis

AD	Microsoft: Active Directory
BCL	Base Class Library
CCC	SAP: Customer Competence Center
CIL	Common Intermediate Language
CLR	Common Language Runtime -
COM	Microsoft: Component Object Model
DLL	Microsoft: Dynamic Link Library
DNS	Directory Name Service
ERP	Enterprise Resource Planning
GUID	Globally Unique Identifier
IIS	Microsoft: Internet Information Services
ITIL	Information Technology Infrastructure Library
MSDN	Microsoft Developer Network
OASIS	Organization for the Advancement of Structured Information Standards
PKI	Public-Key-Infrastruktur
RPC	Remote Procedure Call
SCOM	Microsoft: System Center Operations Manager
SDK	Software Development Kit
SOAP	Ursprünglich: Simple Object Access Protocol. Offiziell seit Version 1.2 nicht mehr als Akronym gebraucht
TAP	Technical Adoption Partner
UDDI	OASIS: Universal Description Discovery and Integration
URI	Uniform Ressource Identification
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WCF	Windows Communication Foundation

WMI	Microsoft: Windows Management Instrumentations
WSDL	Web Service Description Language
WSE	Microsoft: Web Service Enhancements
XML	eXtensible Markup Language

Versicherung über die Selbständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) bzw. §24(4) bzw. §25(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den 13.03.2008 _____

Ort, Datum Unterschrift