Sebastian Frank

Entwurf einer modularen Steuerung für Nanopositionier- und Nanomessmaschinen

Entwurf einer modularen Steuerung für Nanopositionier- und Nanomessmaschinen

Von Sebastian Frank

Universitätsverlag Ilmenau
2007

Impressum

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Diese Arbeit hat der Fakultät Maschinenbau als Dissertation vorgelegen

Tag der Einreichung:	16. Mai 2007
1. Gutachter:	Univ.-Prof. Dr.-Ing. habil. Mathias Weiß, TU Ilmenau
2. Gutachter:	Univ.-Prof. Dr.-Ing. habil. Gerhard Linß, TU Ilmenau
3. Gutachter:	Univ.-Prof. Dr.-Ing. habil. Hendrik Rothe, Universität der Bundeswehr Hamburg
Tag der Verteidigung:	20. September 2007

Technische Universität Ilmenau/Universitätsbibliothek
Universitätsverlag Ilmenau
Postfach 10 05 65
98684 Ilmenau
www.tu-ilmenau.de/universitaetsverlag

Herstellung und Auslieferung
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
www.mv-verlag.de

ISBN 978-3-939473-23-7
urn:nbn:de:gbv:ilm1-2007000238

Kurzfassung

Nanopositionier- und Nanomessmaschinen (NPM-Maschinen) sind Werkzeuge, die den Zugang zum Nanometerbereich ermöglichen. Durch die spezielle Konstruktion der Maschinen ist es möglich, auch in großen Arbeitsbereichen (Ziel: 350 x 350 x 50 mm^3), Objekte mit einer Genauigkeit von wenigen Nanometern zu positionieren und mithilfe geeigneter Messsysteme zu vermessen. Die denkbaren Einsatzgebiete sind dabei nicht auf die Vermessung nanostrukturierter Bauteile begrenzt, sondern dringen durch die Entwicklung geeigneter Nanowerkzeuge auch in den Bereich der gezielten Bearbeitung kleinster Objekte vor.

Durch den Einsatz geeigneter Programme werden die Fachleute auf dem Gebiet der Messtechnik in die Lage versetzt, die Leistungsfähigkeit der NPM-Maschine auszunutzen, ohne selbst zu Softwarespezialisten werden zu müssen. Die vorliegende Dissertation zeigt die Ergebnisse des Entwurfs einer modularen Steuerung für Nanopositionier- und Nanomessmaschinen. Die wichtigsten Ergebnisse der Dissertation sind zunächst der vollständige Entwurf eines NPM-Maschinenbefehlssatzes auf der Basis des Datenformates XML. Dieser spezielle Datensatz bildet die Grundlage für einen grafischen Programmeditor zur Erstellung von Messprogrammen für NPM-Maschinen. Der Bediener kann mithilfe des Editors auf grafischem Wege, unterstützt durch entsprechende Dialoge zur Parametrierung der Befehle, umfangreiche Messprogramme erstellen. Diese werden ebenfalls in Form von XML-Dateien gespeichert. Die Abarbeitung der Messprogramme wird von einem neu programmierten Interpreter realisiert. Dieser dient der Steuerung des Programmablaufes und der Überwachung von Fehlerzuständen. Ein weiterer wichtiger Teil der Dissertation beschäftigt sich mit den Teleservicefunktionen für NPM-Maschinen. Die Ergebnisse zahlreicher Untersuchungen zu Technologien zur Realisierung verteilter Anwendungen führten zur Entwicklung eines Webdienstes, der der Maschinendiagnose dient. An einem weiteren Beispiel in Form einer Handsteuerung für die NPM-Maschine wurden die Möglichkeiten der Fernsteuerung und die Echtzeitvisualisierung des Maschinenzustandes demonstriert.

Für alle aufgeführten Gebiete werden die damit im Zusammenhang stehenden Technologien ausführlich beschrieben.

Abstract

Nanopositioning and nanomeasuring machines (NPM machines) are tools that can make manipulation at nanometer scales possible. Despite the large operational area of the specially constructed machines (the goal is 350mm x 350mm x 50mm), it is possible to position objects with an accuracy of only a few nanometers and to measure their dimensions to a similar resolution. The possible applications are not limited to the measurement of nano-structures; for instance, it is possible to manipulate very tiny object through the application of these nano-tools.

By using software developed for the NPM machines, measurement specialists will be able to use the full capabilities of the machines without having to be become software specialists themselves. This paper describes the design and implementation of a modular control system for nanopositioning and nanomeasuring machines. Fundamental to the design of the system is the development of an NPM "instruction set", implemented using XML syntax, to direct the operation of the control software of the NPM machines. The XML data set is created using a graphical user interface-based editing program developed specifically to facilitate the use of NPM machines in measurement applications. With the aid of the editor, the operator can develop comprehensive measurement programs in an intuitive and easy to use manner. To facilitate the use of the graphical editor, numerous dialogs are available to assist in providing the necessary parameters to the instructions to be executed by the NPM machines. The resultant programs based on the NPM "instruction set" are also stored in XML, and are processed by an interpreter, developed for that purpose, that controls program flow and handles exceptions. This paper also deals with the issue of network control of NPM machines. The results of extensive research into the feasibility of the implementation of distributed applications led to the development of a web-service for the remote management of the operation of NPM machines. The feasibility of remote control was proven through the development of program that allows for manual control of NPM machines. The program also demonstrated the feasibility of real-time visualization of the state of NPM machines, in particular the position of the table.

For the entire subject described, the required technologies will be described in detail.

Vorwort des Verfassers

Mit der Einrichtung des von der DFG geförderten Sonderforschungsbereiches 622 „Nanopositionier- und Nanomessmaschinen“ (NPM-Maschinen) wurde der Forschungsschwerpunkt an der Technischen Universität Ilmenau nachhaltig in Richtung Nanotechnologie beeinflusst. Als wissenschaftlicher Mitarbeiter im Fachgebiet Rechneranwendung im Maschinenbau, unter Leitung von Herrn Univ.-Prof. Dr.-Ing. habil. Mathias Weiß, bot sich mir die Gelegenheit, mich aktiv an der Entwicklung der NPM-Maschine beteiligen zu können. Als Teilprojekt C3 war das Fachgebiet unter der Überschrift „Visualisierung und Teleservice“ sowohl mit der Erarbeitung von Programmen zur Visualisierung von Mess- und Maschinenzustandsdaten, als auch mit dem Entwurf einer Steuerung für NPM-Maschinen beschäftigt. Aufgrund dieses Themenschwerpunktes entwickelte sich die Idee zu einer modularen, fernbedienbaren Steuerung für NPM-Maschinen auf der Basis moderner Technologien wie XML und Webdienste.

Unterstützung bei der Konkretisierung der Zielstellung für die vorliegende Arbeit fand ich bei Herrn Prof. Weiß, der sich auch bereit erklärte, diese als Doktorvater zu betreuen. Während der Bearbeitung des Themas konnte ich stets auf seine konstruktive Kritik und seine uneingeschränkte Unterstützung vertrauen. Dadurch hat er mir immer wieder neue Denkanstöße gegeben und meine Motivation gefördert. Dafür gilt ihm mein besonderer Dank.

Ausdrücklich möchte ich auch Herrn Professor Linß und Herrn Professor Rothe für die Bereitschaft zur Übernahme der Berichterstattung danken.

Aber auch für die zahlreichen fachlichen Diskussionen, die oftmals ganz neue Aspekte eröffneten, möchte ich mich bei Frau Dr. Braunschweig und den anderen Kollegen des Fachgebietes und der Fakultät für Maschinenbau bedanken.

Rückhalt und stetige Ermutigung habe ich während der gesamten Bearbeitungszeit auch durch meine Familie und Freunde erfahren. Nicht zuletzt gilt deshalb auch ihnen mein Dank.

Sebastian Frank Gotha, im Mai 2007

Inhaltsverzeichnis

1 Einleitung

In den letzten Jahren taucht immer häufiger der Begriff „Nanotechnologie“ auf. In immer mehr Bereichen sowohl des technischen als auch des täglichen Lebens begegnen uns Schlagworte, die sich auf Prozesse, Objekte oder Stoffe beziehen, deren metrologische Größenordnung im Bereich von weniger als 100 nm liegt.

Gerade auf dem Gebiet der Halbleiterindustrie hat die Nanotechnologie eine große Bedeutung. Die fortschreitende Miniaturisierung bei der Produktion von Computerchips dringt dabei immer weiter in Bereiche kleinster Abmessungen ein. So sind Strukturbreiten von 90 nm und 65 nm Stand der Technik. Ein Übergang zur 45-nm-Technologie steht in naher Zukunft bevor [1].

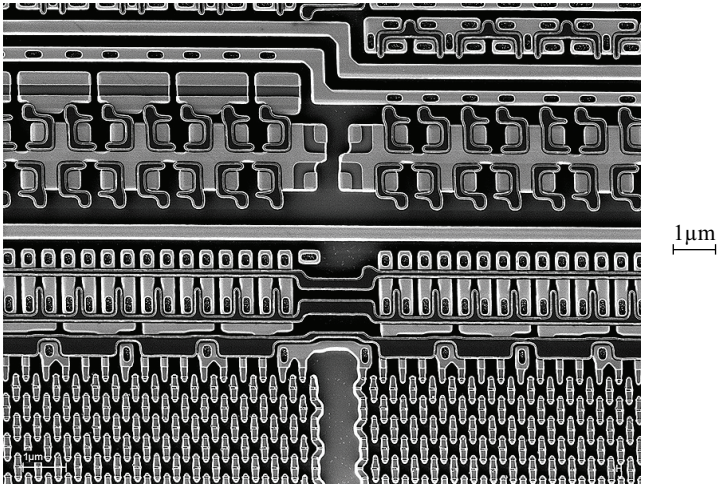


Bild 1: REM-Aufnahme eines DRAM-Schaltkreises, Quelle: Carl Zeiss NTS GmbH
(http://www.smt.zeiss.com/C1256E4600307C70/GraphikTitelIntern/Semicon3gross/$File/semicon3_gross.jpg)

Aber auch im Maschinenbau, der chemischen Industrie, der Werkstofftechnik und sogar in der Lebensmittelindustrie lassen sich Einsatzgebiete für Nanotechnologie finden.

Wie soll der Mensch aber mit derart kleinen Objekten umgehen? Der Nanometerbereich entzieht sich den Sinneswahrnehmungen des Menschen. Nur mithilfe geeigneter Werkzeuge

können Objekte des Nanometerbereiches erstellt, manipuliert, vermessen und visualisiert werden.

Die Nanopositionier- und Nanomessmaschine stellt ein Werkzeug für den Zugang des Menschen zum Nanometerraum dar. Es handelt sich bei dieser Maschine um ein komplexes Mess- und Positioniersystem, welches an der Technischen Universität Ilmenau entwickelt wurde. Die Nanopositionier- und Nanomessmaschine (NPM-Maschine) erlaubt die Positionierung eines Messobjektes mit einer Genauigkeit von weniger als 10 nm. Durch die Kombination mit einem weiteren Messsystem kann das Objekt angetastet und damit beispielsweise die Topografie der Objektoberfläche ermittelt werden. Durch die hohe Genauigkeit und den für die erreichbare Auflösung relativ großen Arbeitsbereich stellt die NPM-Maschine ein wichtiges Werkzeug für unterschiedlichste Anwendungsgebiete dar.

Neben den mechanischen und optoelektronischen Komponenten der NPM-Maschine besteht die Maschine aus einer Vielzahl elektronischer Baugruppen. Diese dienen vor allem der Antriebssteuerung und Lageregelung sowie der Messwertermittlung. Die Steuerung der Maschine erfolgt über einen Bedien-PC. Dafür ist in der Maschine ein USB-Modul vorhanden, welches den Anschluss der Maschine an einen handelsüblichen PC ermöglicht. Die für die Steuerung der NPM-Maschine eingesetzte Software muss den Bediener in geeigneter Art und Weise bei seiner Arbeit unterstützen. Daher werden an die Steuerungssoftware von NPM-Maschinen besondere Anforderungen gestellt.

Die vorliegende Dissertation leistet einen Beitrag für die Verbesserung der Steuerung von Nanopositionier- und Nanomessmaschinen. Dafür wird untersucht welche Komponenten bei der Arbeit mit NPM-Maschinen notwendig sind und wie diese zu einem flexiblen, modular aufgebauten Steuerungsprogramm zusammengefügt werden können. Es wird dargelegt, welche Technologien dafür eingesetzt werden können. Den Nachweis über die Realisierung einzelner Komponenten liefern die im Rahmen der Dissertation entstandenen Programme.

Am Anfang der Arbeit wird zunächst der Aufbau von NPM-Maschinen dargestellt. Es wird kurz auf die mechanischen, optoelektronischen und elektronischen Baugruppen eingegangen. Weiterhin wird gezeigt, wie der Informationsfluss sowohl innerhalb als auch zwischen Maschine und Bedien-PC organisiert ist.

Darauf aufbauend wird in Kapitel 3 die Aufgabenstellung konkretisiert. Es wird aufgezeigt, welche Arbeitsvorgänge im Zusammenhang mit NPM-Maschinen auftreten können und

welche Programmteile zur Bewältigung der einzelnen Arbeitsschritte notwendig sind. Einige dieser Programme wurden bereits realisiert, andere sind Gegenstand zukünftiger Arbeiten.

Nach einer kurzen Übersicht über eine Auswahl vorhandener Software zur Steuerung von Werkzeug- und Messmaschinen werden in den Kapiteln 5, 6 und 7 die Programmteile und Technologien beschrieben, die im Zusammenhang mit dem Entwurf einer modularen Steuerung von Nanopositionier- und Nanomessmaschinen entwickelt wurden. Die wichtigsten Arbeiten an der Verwirklichung dieses Zieles waren die Überführung des Maschinenbefehlssatzes in das XML-Format, die Entwicklung eines grafischen Programmeditors für die Erstellung von Messprogrammen, die Programmierung eines Diagnosemoduls und die Realisierung von Teleservicefunktionen für die Steuerung der Maschine über große Entfernungen hinweg.

Abschließend wird in einem kurzen Ausblick auf die Ziele zukünftiger Arbeiten eingegangen. Dazu zählen insbesondere Werkzeuge zur Visualisierung und Archivierung von Messergebnissen als auch die Weiterentwicklung des Editors und des Interpreters, der die Abarbeitung von Messprogrammen und die Verarbeitung von Fehlersituationen realisiert.

Der Entwurf einer modularen Steuerung für Nanopositionier- und Nanomessmaschinen ist ein umfangreiches Themengebiet. Die Dissertation wird im Folgenden einige Aspekte aus diesem Bereich konkretisieren.

2 Nanopositionier- und Nanomessmaschinen

Die Nanopositionier- und Nanomessmaschine ist ein komplexes System zur Messung und Bearbeitung von Objekten im Nanometerbereich. An der Technischen Universität Ilmenau wurde im Fachgebiet von Herrn Professor Gerd Jäger an der Entwicklung einer solchen Maschine gearbeitet. Der verfügbare Arbeitsbereich einer ersten, in Zusammenarbeit mit der Firma SIOS Messtechnik Ilmenau, gebauten Maschine betrug 25 x 25 mm² in der Fläche und 5 mm in vertikaler Richtung. Die Auflösung der Maschine beträgt 0,1 nm. Für die kombinierte Messunsicherheit kann nach [21] ein Wert von 6,62 nm in Richtung der X- und Y-Achsen und ein Wert von 7,16 nm in Richtung der Z-Achse angegeben werden [46].

Aufgrund der hohen Auflösung und der geringen Messunsicherheit ist die Nanopositionier- und Nanomessmaschine für viele Anwendungsfälle von großem Interesse. Nachteilig wirkt sich aber der relativ kleine Arbeitsbereich aus. Deshalb wird intensiv an einer Erweiterung des Arbeitsbereiches auf zunächst 200 x 200 x 5 mm³ gearbeitet und mittelfristig ein Bereich von 350 x 350 x 50 mm³ angestrebt. Die Forschungsanstrengungen zur Überwindung der dabei auftretenden technologischen Schwierigkeiten führten zur Einrichtung eines Sonderforschungsbereiches „Nanopositionier- und Nanomessmaschinen, SFB 622“, dessen Ziel es ist die Grundlagen zur Realisierung einer derartigen Präzisionsmaschine zu erforschen. Dazu wurden innerhalb des SFB eine Vielzahl von Forschungsdisziplinen vereinigt.

Um das komplexe System Nanopositionier- und Nanomessmaschine vorzustellen, soll zunächst der mechanische Grundaufbau der NPM-Maschine erläutert werden. Im Anschluss daran erfolgt eine Übersicht über die elektronische Steuerung der Maschine und den sich daraus ableitenden Informationsfluss innerhalb des Messsystems NPM-Maschine.

2.1 Der Aufbau einer NPM-Maschine

Um genaue Messungen durchführen zu können, kommt es darauf an die Messunsicherheit des Messsystems zu minimieren. Gerade in den metrologischen Dimensionen des Nanometers wirken sich selbst kleine Fehler gravierend aus. In [21] werden in Kapitel 2 einige Geräte beschrieben, die den Stand der Technik in der Messtechnik für nanoskalige Größen repräsentieren. Allen Geräten ist eigen, dass sie sowohl durch Anordnung von Messsystem

und Messobjekt zueinander als auch durch den Einsatz unterschiedlichster Messsysteme versuchen den systematischen Messfehler so gering wie möglich zu halten.

Den größten Einfluss auf das Messunsicherheitsbudget in der Koordinatenmesstechnik hat die Anordnung von Messobjekt und Messsystem zueinander. Bereits Ernst Abbe (1840 – 1905) beschrieb die Wirkung von Kippfehlern erster Ordnung auf das Messergebnis. Als ideale Anordnung von Prüfling und Messgerät beschreibt er die fluchtende Anordnung beider. Dieses Prinzip stellt das nach ihm benannte *Abbesche Komparatorprinzip* dar. Eine wichtige Anforderung an eine genaue Messung ist daher, das Abbesche Komparatorprinzip einzuhalten, um den Einfluss von Kippfehlern erster Ordnung zu minimieren.

In [45] werden verschiedene Designvarianten vorgestellt, die das Komparatorprinzip in mindestens einer Messachse realisieren. Dazu zählen:

Koordinatenmessgeräte

Etablierte Koordinatenmessgeräte arbeiten nach dem Prinzip des ortsfesten Messobjektes und der Antastung mittels eines in drei Achsen beweglichen Antastelements. Die Position des Antastpunktes in X- und Y-Richtung wird durch Linearmaßstäbe in den Bewegungsachsen gemessen. Daraus resultiert ein deutlicher Versatz zum Messpunkt (siehe Abstände a und b in Bild 2), der eine Verkippung nach sich ziehen kann. Das Abbesche Komparatorprinzip wird bei diesen Geräten nur in der vertikalen Achse realisiert [21],[54].

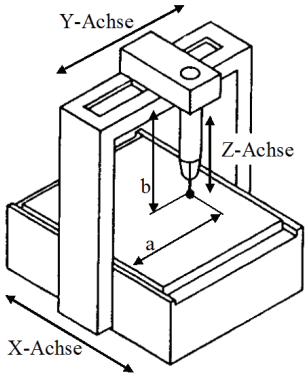


Bild 2: Aufbau eines Koordinatenmessgerätes in Portalbauweise, nach [54] Seite 274

Tastschnittgräte

Für die Bestimmung von Oberflächentopologien haben sich in den letzten Jahren verschiedene Hersteller auf die Produktion von Tastschnittgeräten mit größeren

Messbereichen spezialisiert [45]. Bei diesen Geräten wird über einen Tasthebel eine Antastung in vertikaler Richtung vorgenommen. Auch in dieser Anordnung sind Kippfehler zu erwarten, da hier das Komparatorprinzip nicht in allen drei Raumachsen gleichzeitig realisiert werden kann. Zusätzlich ist zu beachten, dass sich der Messtaster auf einer Kreisbahn bewegt [21].

Rasterkraftmikroskop Veritekt-3

Ein modifiziertes Rasterkraftmikroskop vom Typ Veritekt-3 der Firma Zeiss wird in [21] als Vorläufer der Nanopositionier- und Nanomessmaschine beschrieben. Hier ist es erstmals gelungen, durch Verwendung einer Spiegelecke und drei interferometrischen Messsystemen, das Abbesche Komparatorprinzip in allen drei Messsachsen einzuhalten. Hierbei wird das Messobjekt bewegt und von einem ortsfesten Antastsystem angetastet. Die optischen Messsysteme treffen sich in einem virtuellen Schnittpunkt, der mit dem Antastpunkt zusammenfällt. In Bild 3 ist der Aufbau des beschriebenen Rasterkraftmikroskops zu sehen.

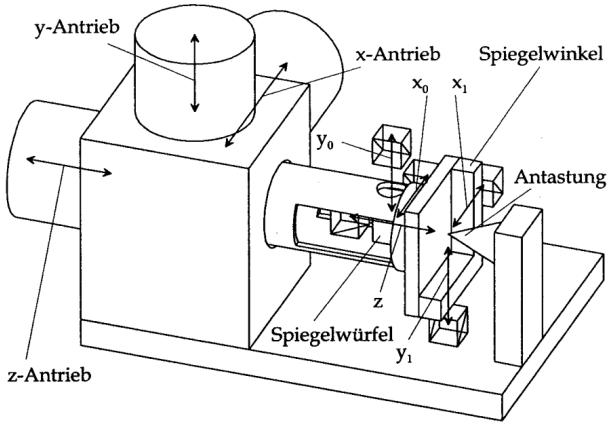


Bild 3: Rasterkraftmikroskop Veritekt-3, Quelle: [21]

2.1.1 Mechanischer Aufbau der Nanopositionier- und Nanomessmaschine

Um die Anforderungen an die Genauigkeit einer Messung im Nanometerbereich realisieren zu können, ist neben geeigneten Messsystemen auch deren Anordnung von zentraler Bedeutung. Hausotte schreibt in [21] Kapitel 3.3:

> „[...] Aus dem Vergleich der [...] Designvarianten wird ersichtlich, dass es hierfür [Anm.: Forderung nach dem Komparatorprinzip in allen drei Achsen] nur eine einzige Grundanordnung gibt. In dieser Anordnung sind das Antastsystem und die drei

Längenmesssysteme gemeinsam an einem stabilen Rahmen befestigt. Die Messachsen der Längenmesssysteme müssen sich in dem Kontaktpunkt des Antastsystems schneiden [...]“

Aus dieser Forderung heraus wurde für die Nanopositionier- und Nanomessmaschine der in Bild 4 dargestellte Grundaufbau entwickelt. Dabei nimmt ein gestellfester Zerodurrahmen die drei Längenmesssysteme für die Position der Spiegelecke (Interferometer für X-, Y- und Z-Achse) und die beiden Winkelsensoren zur Messung der Verkippung der Spiegelecke um die X- und Y-Achse auf. Weiterhin stellt der Zerodurrahmen Befestigungspunkte für das Antast- bzw. Messsystem zur Verfügung. Die Verwendung von Zerodur für den Gestellrahmen soll thermische Einflüsse auf die Lage der Messsysteme zueinander minimieren.

Die Spiegelecke ermöglicht die Bestimmung des Antastpunktes als Wegmessung über die drei Interferometer der Messachsen. Im Idealfall besteht die Spiegelecke aus drei orthogonal aufeinander stehenden Spiegelflächen.

Für die Relativbewegung zwischen Messobjekt und Messsystem kommt in dieser Variante nur das Prinzip des bewegten Objektes bei ortsfestem Messsystem in Frage. Dazu wird der gesamte Messtisch, auf dem die Spiegelecke befestigt ist, bewegt. Für die Antriebe und Führungen in X- und Y-Richtung sowie für die Z-Achse werden unterschiedliche Antriebe und Lager verwendet. Die Eigenschaften der Antriebe und Führungen haben direkten Einfluss auf das Messergebnis, sodass Optimierungen dieser Komponenten intensiver Forschungsgegenstand des an der Technischen Universität Ilmenau angesiedelten Sonderforschungsbereiches 622 sind.

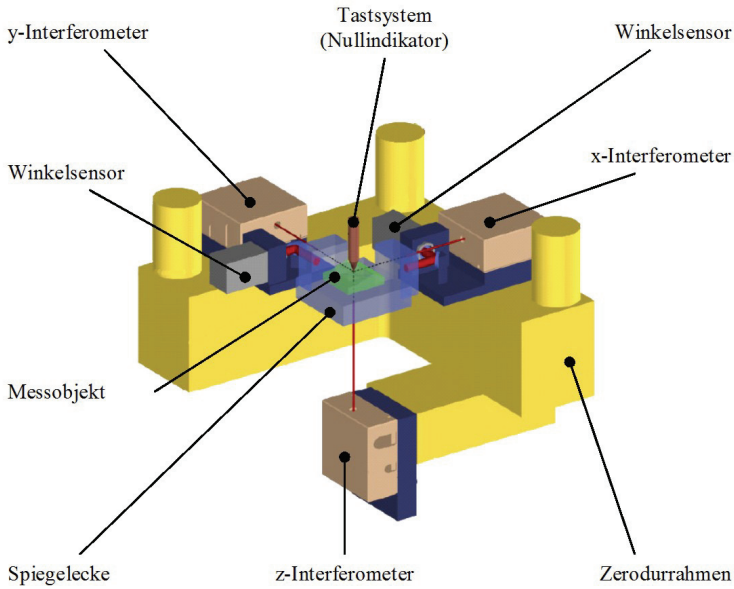


Bild 4: Grundaufbau der NPM-Maschine

Zum Zeitpunkt der Erstellung der vorliegenden Dissertation wurden folgende Antriebe und Führungen verwendet [21]:

- Die Antriebe für die X- und Y-Achse sind zwei identische elektromagnetische Linearantriebe der amerikanischen Firma *Kimco Magnetics*. Beide Antriebe sind orthogonal zueinander angeordnet.
- Der Antrieb in Z-Richtung erfolgt mithilfe von vier zylindrischen Linearantrieben, ebenfalls von der Firma Kimco Magnetics.
- Die Führung des Antriebssystems in X- und Y-Richtung wird jeweils durch eine Wälzkörperführung der Firma MICOS realisiert.
- Die Führung in Z-Richtung erfolgt durch drei Zylinderführungen.

Das gesamte Antriebs- und Führungssystem zeigt Bild 5. An der Oberseite des Systems ist die Aufnahme für die Messspiegelecke zu erkennen.

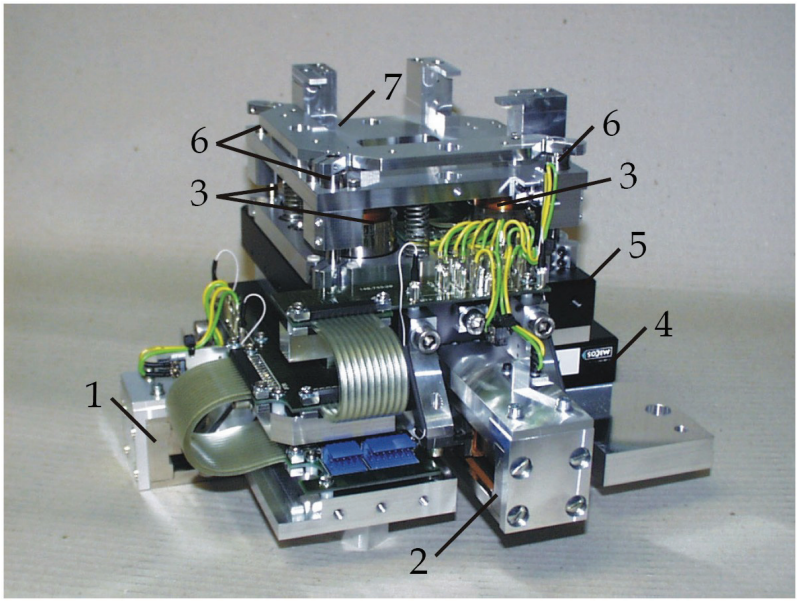


Bild 5: Antriebs- und Führungssystem der NPM-Maschine [21]: 1 X-Antrieb, 2 Y-Antrieb, 3 Z-Antriebe, 4 X-Führung, 5 Y-Führung, 6 Z-Führungen, 7 Aufnahme für Spiegelecke

Die Verkippung der Spiegelecke lässt sich durch unterschiedliche Ansteuerung der vier Linearantriebe in Z-Richtung kompensieren. Um die Führungsabweichung der Wälzführungen in X- und Y-Richtung zu berücksichtigen, wurden die Abweichungen vermessen.

Für die Gewährleistung des messtechnischen Grundsatzes nach Abbe ist es erforderlich, dass Antastpunkt und Maßverkörperung über den gesamten Messbereich fluchtend angeordnet sind. Um dies in allen drei Raumkoordinaten zu ermöglichen, muss die Spiegelecke in ihren lateralen Ausdehnungen größer als der Messbereich der NPM-Maschine sein. Nur so ist sichergestellt, dass sich die Messstrahlen der Interferometer virtuell im Antastpunkt schneiden. Die durch den derzeitigen Messbereich von 25 x 25 x 5 mm³ resultierenden „großen“ Spiegelflächen können aus Gewichts-, Kosten- und Fertigungsgründen nur aus Einzelteilen zusammengesetzt werden. Die durch Fertigung und Montage resultierenden Abweichungen von der Idealform müssen durch interferometrische Vermessung der Spiegelecke berücksichtigt werden. Die durch die Messung gewonnene Oberflächentopologie der Spiegelecke geht in die Berechnung des Messwertes ein. Vergleichsmessungen haben eine Langzeitstabilität der Spiegelecke nachgewiesen.

Für die Längenmessung der Spiegelposition kommen Interferometer der Firma SIOS zum Einsatz [45]. Um den Wärmeeintrag in das Messsystem zu minimieren, wird die Erzeugung der für die Interferometer erforderlichen Laserstrahlung in das externe Versorgungs- und Auswertegerät ausgelagert (siehe 2.1.2).

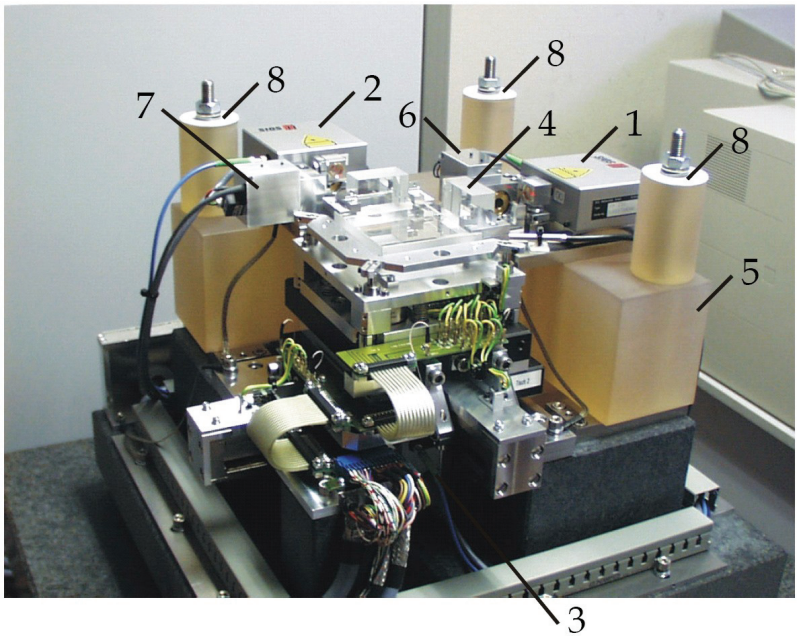


Bild 6: NPM-Maschine [24]: 1, 2, 3 Interferometer X-, Y-, Z-Achse, 4 Spiegelecke, 5 Zerodurkörper, 6,7 Winkelsensoren, 8 Befestigungspunkte für Antastsystem

Aufgrund der vielfältigen Einsatzgebiete der NPM-Maschine sind verschiedenste Mess- und Bearbeitungssysteme denkbar. Darunter sind Antastsysteme wie der Autofokussensor der Firma SIOS, Weißlichtinterferometer oder auch Rasterkraftsensoren. Im Rahmen des SFB 622 wird darüber hinaus an verschiedenen Sensoren mit mehreren Tastspitzen geforscht. Als Bearbeitungstools können Mikrogreifer oder Elektronenstrahlquellen zum Einsatz kommen. Die einzelnen Mess- und Bearbeitungssysteme können an den in Bild 6 ersichtlichen Befestigungspunkten (Ziffer 8) angebracht werden. Die im Zusammenhang mit dem SFB 622 entwickelten Prototypen verwenden als Messsystem den Autofokussensor der Firma SIOS [23],[25].

2.1.2 Gesamtsystem

Neben den mechanischen Komponenten besteht eine Nanopositionier- und Nanomessmaschine aus einer Vielzahl unterschiedlicher elektronischer und

optoelektronischer Komponenten. Diese sind in einer Versorgungs- und Auswertungseinheit zusammengefasst. Eine genaue Beschreibung der einzelnen Komponenten ist in [21] nachzulesen. An dieser Stelle soll lediglich ein kleiner Überblick über die angesprochenen Baugruppen erfolgen.

Lasereinheit

Um den Wärmeeintrag in der Nähe der messtechnisch sensiblen Baugruppen zu minimieren, werden die Laserinterferometer über Lichtleitkabel von außen mit frequenzstabilisierter Laserstrahlung versorgt. Die Lasereinheit besteht aus drei einzelnen HeNe-Lasern. Durch eine Frequenzregelelektronik wird die Resonatorlänge der Laser variiert und somit eine hohe Frequenzstabilität erreicht [21].

Interferometereinheit

Die Interferometereinheit dient der Vorverarbeitung der Interferometer-, Winkelmess- und Umweltsignale. Durch entsprechende Eingangsmodule und Verstärker werden diese Signale so aufbereitet, dass sie in der anschließenden DSP-Einheit weiter verarbeitet werden können. Die Interferometer- und Winkelmesssignale werden direkt als analoge Spannungswerte an den DSP geführt. Die Umweltsignale Luftdruck und Lufttemperatur gehen in die Korrekturalgorithmen des DSPs zur Berechnung der Positionswerte ein. Da diese Werte keine große Dynamik besitzen, ist eine Verarbeitung mit niedrigerer Priorität ausreichend. Aus diesem Grund werden diese Signale bereits auf der Interferometerkarte digitalisiert und anschließend über eine serielle Schnittstelle an den DSP gesendet [21].

DSP-Einheit

Die DSP-Einheit (DSP – Digitaler Signal-Prozessor) besteht aus einem DSP-Modul, einem USB-Modul, drei Tachocontroller-Modulen, drei AD/DA-Wandler-Modulen sowie der notwendigen Netzteilbaugruppe. Der auf dem DSP-Modul verwendete Prozessor ist ein Gleitkommaprozessor der Firma Texas Instruments mit einer Taktfrequenz von 60 MHz. Die Tachocontroller-Module in Verbindung mit den AD/DA-Wandlern ermöglichen die Bestimmung des Messwertes für die Position der Winkelspiegelecke in den drei Raumkoordinaten. Über ein Bussystem sind die Tachocontroller-, die AD/DA-Wandler-Module und der DSP miteinander verbunden. Mithilfe des USB-Moduls können die ermittelten Messwerte zur Auswertung an einen PC übertragen werden. Über die USB-Schnittstelle erfolgt ebenfalls die Steuerung der Maschine [21].

Motoreinheit

Die für die Antriebe der NPM-Maschine notwendigen Antriebsströme werden von Leistungsverstärkern bereitgestellt. Diese sind in der Motoreinheit angeordnet und werden durch analoge Signale der DA-Wandler angesteuert. Gleichzeitig sendet die Motoreinheit Fehler- und Endlagensignale an den DSP [21].

Bild 7 zeigt das NPM-Gesamtsystem – bestehend aus der Maschine, der Versorgungseinheit und dem Bedien-PC.

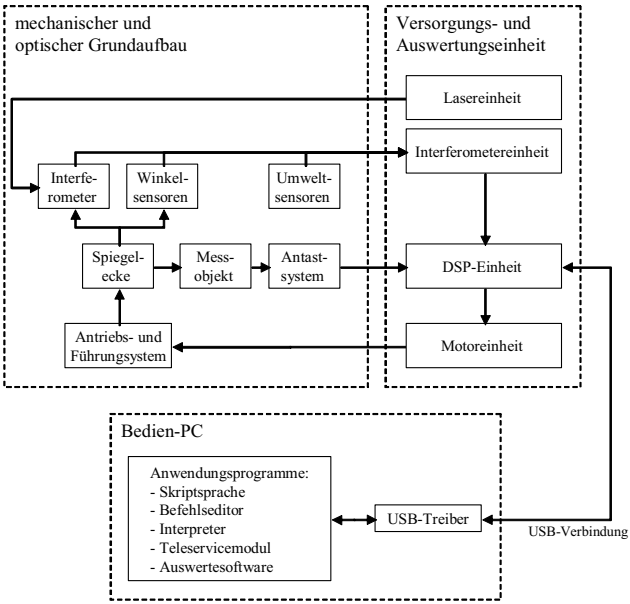


Bild 7: Gesamtaufbau der NPM-Maschine mit Versorgungseinheit und Bedien-PC, nach [21]

Die Bedienung und Steuerung der NPM-Maschine erfolgt vom Bedien-PC aus. Dieser ist über eine USB-Schnittstelle mit der Versorgungseinheit der Maschine und damit mit dem DSP verbunden. Für die Kommunikation zwischen PC und DSP über USB existiert ein spezieller Treiber. Dieser übernimmt die eigentlichen Aufgaben der Datenübertragung und stellt dazu entsprechende Funktionen bereit. Die Kommunikation mit dem DSP wird dadurch erleichtert. Die Grundlagen der Kommunikation mit der NPM-Maschine werden im folgenden Abschnitt erläutert.

2.2 Die Steuerung einer NPM-Maschine

Im Informationsfluss der NPM-Maschine existieren unterschiedlichste Signalströme. So liefern die Interferometer und die Winkelsensoren Signale für die Lage und Position der Winkelspiegelecke. Die Umweltsensoren liefern Aussagen über Temperatur und Luftdruck. Die Endlagenschalter melden die Grenzen des Arbeitsbereiches in den entsprechenden Raumachsen und schließlich liefern die verwendeten Antastsysteme entsprechende Messwerte über das Messobjekt. Gleichzeitig müssen von den Motortreibern entsprechende Stellgrößen an die Antriebe ausgegeben werden. Alle diese Aufgaben werden durch die entsprechenden mechanischen bzw. elektronischen Komponenten realisiert. Die einzelnen Komponenten stehen dazu über ein Bussystem innerhalb der Versorgungseinheit in Verbindung. Den zentralen Punkt innerhalb der Versorgungseinheit bildet der DSP mit der entsprechenden Software. Ein weiterer wichtiger Teil der DSP-Software ist für die Kommunikation zwischen NPM-Maschine und Bedien-PC verantwortlich. Einen Überblick über die einzelnen Schichten der Informationsströme zeigt Bild 8.

Die Steuerung der Maschine erfolgt vom Bedien-PC aus. Über die USB-Schnittstelle werden Befehle an die NPM-Maschine gesendet bzw. Daten aus der Maschine gelesen. Dazu ist der in der DSP-Software implementierte Funktionsumfang zu verwenden. Über die vom USB-Treiber bereitgestellten Funktionen kann auf die DSP-Software zugegriffen werden.

Für die Steuerung der entwickelten NPM-Maschinen stehen folgende Funktionen bereit:

- Funktionen zum Auslesen von Maschineninformationen
- Funktionen zum Einstellen von Maschinenparametern
- Funktionen zum Steuern der Antriebsachsen
- Funktionen zum Ausführen von Messoperationen
- Funktionen für den Datenaustausch
- Funktionen zum Auslesen von Fehlerzuständen
- Funktionen zur Steuerung von Bearbeitungstools (zukünftig)

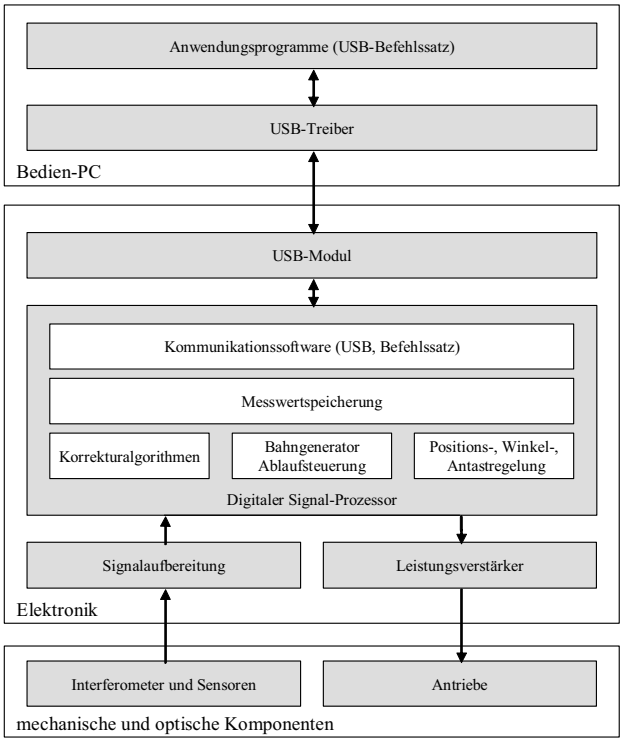


Bild 8: Informationsfluss der NPM-Maschine

Im Folgenden wird das Zusammenspiel von USB-Treiber und Befehlssatz der Maschine näher beschrieben.

2.2.1 USB-Schnittstelle und Treiber

Die Steuerung der NPM-Maschine sollte ohne spezielle Zusatzhardware möglich sein. Ein handelsüblicher PC sollte für die Bedienung genügen. Dazu musste eine geeignete Schnittstelle zwischen der elektronischen Versorgungseinheit der NPM-Maschine und dem Bedien-PC gewählt werden. Die vom DSP bereitgestellte serielle RS-232-Schnittstelle scheidet aufgrund der zu geringen Datenübertragungsrate für die Kommunikation zwischen Bedien-PC und NPM-Maschine aus. Ein direkter Anschluss an das Bussystem der Versorgungseinheit ist ohne zusätzliche Hardware im PC ebenfalls nicht möglich. Stand der Technik ist der Universal Serial Bus (USB). Aufgrund seiner anwenderfreundlichen Bedienung und der ausreichenden Übertragungsrate (Stand der beschriebenen Maschine: USB 1.1 = 12 MBit/s) wurde die Entscheidung zugunsten dieser Schnittstelle getroffen.

Zusätzliche Hardware ist nur auf der Seite der NPM-Maschine, in Form eines USB-Controllers innerhalb der Versorgungseinheit, notwendig.

Dazu wurde die Versorgungseinheit um einen USB-Controller der Firma Philips erweitert [21]. Der PDIUSB12D entspricht dem USB-Standard 1.1 und verfügt über ein theoretisches Datentransfervolumen von 9 MBit/s im sogenannten *Bulk-Modus*. Dieser Wert ist für den mit der beschriebenen Hardware erreichbaren Datenstrom ausreichend. In einer späteren Version der DSP-Einheit wird der USB-Controller gegen ein Modul des Standards 2.0 mit theoretischen Übertragungsraten von bis zu 480 MBit/s ausgetauscht. Im Rahmen dieser Dissertation wurde allerdings ausschließlich mit dem 1.1-Controller gearbeitet. Für die Kommunikation zwischen Bedien-PC und Versorgungseinheit hat dies allerdings, abgesehen vom Geschwindigkeitsgewinn, keinen Einfluss, da die vom USB-Treiber bereitgestellten Signaturen (bestehend Funktionsname und Parameterliste) erhalten bleiben.

Der verwendete USB-Treiber stammt ebenfalls von der Firma SIOS Messtechnik Ilmenau. Der Funktionsumfang des Treibers ist für mehrere von der Firma SIOS vertriebene Geräte mit USB-Schnittstelle entwickelt worden. Für die Kommunikation mit der NPM-Maschine ist lediglich eine Teilmenge der insgesamt 47 Funktionen notwendig. Dazu zählen beispielsweise:

- SearchForSIOSDevices — liefert die Anzahl von angeschlossenen Geräten die auf den SIOS-Treiber zugreifen
- CompWriteUSBDevice — sendet Daten (Befehle) an die NPM-Maschine
- CompReadUSBDevice — fordert Daten von der NPM-Maschine an

Darüber hinaus gibt es Funktionen für die Konfiguration, zum Auslesen eines Flash-Speichers des USB-Controllers und zur Abfrage von Fehlerzuständen.

Die hauptsächliche Kommunikation erfolgt anhand der beiden Funktionen CompWriteUSBDevice und CompReadUSBDevice. Sie ermöglichen das Schreiben an bzw. das Lesen von der Maschine. Die Funktionsprototypen beider Funktionen haben in der Programmiersprache C# folgende Struktur:

```
[DllImport("SIOSUSB.DLL", SetLastError=true)]
int CompWriteUSBDevice(int DeviceNum, int BytesToWrite, byte[] InArray);
```

```
[DllImport("SIOSUSB.DLL", SetLastError=true)]
int CompReadUSBDevice(int DeviceNum, byte[] OutArray);
```

Beide Funktionen erwarten als Parameter die Angabe, auf welches SIOS-Gerät zugegriffen werden soll. Die Gerätenummern können anhand der Funktion *SearchForSIOSDevices* in einer Liste gespeichert werden. Alle angeschlossenen Geräte werden durch einen laufenden Index von 0 ... n indiziert. In der Regel ist der Parameter *DeviceNumber* gleich 0.

Im Fall von *CompWriteUSBDevice* liefert der Rückgabewert der Funktion die Anzahl der über den Treiber gesendeten Bytes. Mithilfe eines Vergleichs des Rückgabewertes und der durch den Parameter *BytesToWrite* übergebenen, theoretischen Anzahl kann eine Aussage über den Erfolg des Schreibbefehls getroffen werden. Weichen beide Werte voneinander ab, so war es nicht möglich den Schreib-Befehl erfolgreich zu übermitteln. Eine Fehlerabfrage ist in diesem Fall notwendig.

Der Datenaustausch zwischen Programm und Maschine erfolgt durch ein Bytefeld. Das Bytefeld kann maximal 32767 Bytes groß sein. Sollten mehr Daten im Speicher des DSP vorhanden sein, muss der Lesevorgang entsprechend wiederholt werden. Der Aufbau des Bytefeldes ist von der Funktion abhängig. Für den Schreibbefehl gibt Bild 9 und für den Lesebefehl zeigt Bild 10 einen Überblick über den entsprechenden Aufbau des Feldes.

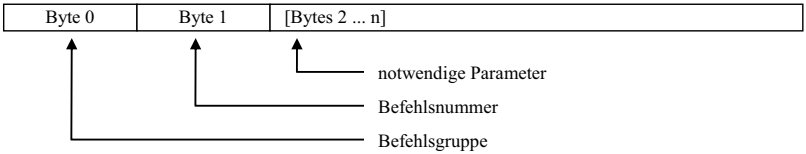


Bild 9: Struktur des Bytefeldes für den Schreibbefehl CompWriteUSBDevice

Die Anzahl der Parameter, die der Schreibbefehl erwartet, hängt von dem gewünschten Maschinenbefehl ab. Es existieren Befehle ohne Parameter. Hier genügt die Angabe von Befehlsgruppe und Befehlsnummer. Anderenfalls existieren Maschinenbefehle mit einer variablen Anzahl an Parametern. Beispielsweise benötigt ein Verfahrbefehl die Angabe von Geschwindigkeit und Koordinaten der Bewegung.

Der Lesebefehl ermittelt die Antwort der Maschine auf den unmittelbar zuvor ausgeführten Schreibbefehl. Dazu werden zu Kontrollzwecken Befehlsgruppe und –nummer nochmals übermittelt. Das dritte Byte (Byte 2) repräsentiert eine Fehlernummer. Sie ist im Fehlerfall ungleich null. Daran schließen sich die eigentlichen Daten an. Hier können beispielsweise die Koordinaten eines gemessenen Punktes oder der Messtischposition übertragen werden.

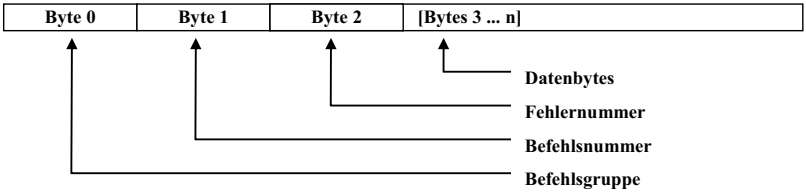


Bild 10: Struktur des Bytefeldes für den Befehl CompReadUSBDevice

Die im DSP der Versorgungseinheit implementierten Maschinenbefehle sind in Gruppen zusammenfasst. Einzelheiten dazu sind im folgenden Kapitel 2.2.2 zusammengefasst.

2.2.2 Befehlssatz der Maschine

Der Befehlssatz der Maschine ist im DSP der Versorgungseinheit hinterlegt. Er steuert alle zentralen, maschinennahen Funktionen innerhalb der „Hardware“ der NPM-Maschine. Der Zugriff auf den Befehlsumfang über den Treiber wurde im vorangegangenen Kapitel beschrieben. Derzeit existieren in der aktuellen DSP-Software 9 Befehlsgruppen mit insgesamt 58 Maschinenbefehlen. Die Befehlsgruppen enthalten im Einzelnen Befehle mit folgenden Funktionen:

- Loopback: Für Testzwecke des Treibers werden alle gesendeten Daten direkt zurückgesendet.
- Grundbefehle: Befehle zum Rücksetzen und zur Datenmanipulation im Adressbereich des DSPs.
- Konfigurationsbefehle:
  Befehle zur Konfiguration von Mess-, Antriebs- und Regelsystemen.
- Mess- und Positionierbefehle:
  Befehle zur Durchführung von Messungen und zur Positionierung des Messtisches. (3 Gruppen)
- Datenaustauschbefehle:
  Befehle zur Steuerung des Datenaustausches zwischen DSP und PC.
- UART-Datenaustauschbefehle:
  Diese Befehle stellen einen direkten Zugriff auf die seriellen Schnittstellen der Interferometereinheiten zur Verfügung.

- Testbefehle: Spezielle Testbefehle um die Funktion der Antriebe zu prüfen.

Mithilfe dieser umfangreichen Befehlssammlung kann der Bediener die gesamten internen Vorgänge steuern und ermittelte Daten sowie Maschinenzustandsinformationen auslesen. Eine Erweiterung des Befehlsumfangs innerhalb der DSP-Software ist jederzeit möglich. Die Struktur der Kommunikation wird aber beibehalten: Jeder Befehl besteht aus Befehlsgruppennummer, Befehlsnummer und einer entsprechenden Anzahl etwaiger Parameter.

2.2.3 Verwendete Software im Bedien-PC

Bisher war der Befehlssatz nur über eine Vielzahl kleiner, im Funktionsumfang eingeschränkter Spezialprogramme zu benutzen. Einige dieser Programme wurden in Borland Delphi entwickelt. Die Erweiterung dieser speziellen Programme ist mit einer Neuprogrammierung und -kompilierung verbunden.

Weiterhin existiert eine Skriptsprache, die den Befehlsumfang in eingeschränktem Maße nutzbar macht. Eine Änderung der DSP-Software ist in diesem Fall mit aufwendigen Erweiterungen der Skriptsprache und umfangreichen Änderungen des nachgeschalteten Interpreters verbunden.

Ein weiterer Ansatzpunkt, um vom PC aus auf die Befehle der Maschine zugreifen zu können, erfolgt über eine Implementierung des USB-Treibers in MATLAB® der Firma MathWorks [33]. Hier werden die Auswertefunktionen der Software dazu genutzt, um eine Auswertung von Messergebnissen vorzunehmen. Allerdings ist auch hier der verfügbare Befehlsumfang eingeschränkt.

Die hier nur kurz beschriebene Vielfalt der zahlreichen Programmumgebungen zur Steuerung der NPM-Maschine zeigt, dass eine Einarbeitung in die Steuerung sehr viel Zeit in Anspruch nimmt und zum Teil den Kauf zusätzlicher Software (z.B. MATLAB®) voraussetzt. Dieser Zustand ist für die Verwendung von NPM-Maschinen innerhalb einer Laborumgebung mit einer begrenzten Anzahl an Mitarbeitern, die eng in die Entwicklung der Maschine eingebunden sind, denkbar. Für die Anwendung der NPM-Maschine als Mess- und Positioniergerät in anderen Branchen ist aber die Entwicklung einer einheitlichen, benutzerfreundlichen Steuerung unumgänglich.

3 Präzisierung der Aufgabenstellung

Der an der Technischen Universität Ilmenau eingerichtete Sonderforschungsbereich 622 „Nanopositionier- und Nanomessmaschinen“ beschäftigt sich intensiv mit der Entwicklung einer NPM-Maschine. Der zu realisierende Arbeitsbereich ist mit 200 x 200 x 50 mm³ deutlich größer als der der im vorangegangenen Kapitel beschriebenen Maschine. Die Zielstellung hat eine Vielzahl ungelöster Probleme hinsichtlich Antriebs- und Führungssystem, Werkstoffauswahl oder Auswahl einer geeigneten Hardwarestruktur für die interne Maschinensteuerung aufgeworfen. Neben der Beherrschung aller technologischen Aspekte zur Entwicklung eines Messsystems für Messaufgaben im Bereich des Nanometers ist es notwendig die Bedienung einer Nanopositionier- und Nanomessmaschine durch ein geeignetes Steuerungsprogramm zu ermöglichen.

Ausgehend von der in den vorangegangenen Kapiteln beschrieben NPM-Maschine müssen alle Arbeitsschritte, die in Zusammenhang mit der Arbeit an NPM-Maschinen stehen, durch entsprechende Softwaremodule unterstützt werden. Dazu zählen:

- Konfiguration der Maschine

  Hierunter fallen Arbeiten im Zusammenhang mit der Einstellung der maschineninternen Parameter z. B. für die Antriebe, die Regler oder die Interferometer. Aber auch die Korrekturfunktionen zum Ausgleich der Fehler der Winkelspiegelecke müssen bei der Konfiguration der Maschine angegeben werden.

- Erstellung von Messprogrammen

  Unter Messprogrammen sollen hier Programme verstanden werden, die auf dem Befehlssatz der Maschine basieren und entsprechende Aktionen, zum Beispiel das Vermessen einer Oberfläche, ausführen. Ein Messprogramm kann aber auch dazu dienen, lediglich den Messtisch an eine bestimmte Position zu fahren, um ein Messobjekt verschiedenen Untersuchungen zuzuführen. In dieser Arbeitsphase müssen die Maschinenbefehle dem

Nutzer in geeigneter Form angeboten und zu Messprogrammen zusammengefügt werden können.

- Abarbeitung von Messprogrammen

  Die mithilfe eines Programmeditors erstellten Programme müssen zunächst auf Gültigkeit geprüft werden. Hierzu zählt beispielsweise, dass geprüft werden muss, ob die Reihenfolge verschiedener Befehle eingehalten wird oder ob Verfahrbefehle ein Verlassen des Arbeitsraumes zur Folge haben. Anschließend werden die Programme durch einen Interpreter abgearbeitet. Der Arbeitsfortschritt muss dem Nutzer in geeigneter Form angezeigt werden. Dazu zählt beispielsweise auch die Echtzeitvisualisierung der gemessenen Werte (Online-Visualisierung). Aber auch die Visualisierung von Fehlermeldungen ist in diesem Arbeitsschritt von großer Bedeutung.

- Auswertung und Archivierung von Messergebnissen

  Durch Messprogramme ermittelte Messwerte müssen in geeigneter Form ausgewertet werden können. In einem ersten Schritt ist dies beispielsweise die Darstellung in einer dreidimensionalen Grafik (Offline-Visualisierung). Aber auch Berechnungsalgorithmen können diese Arbeitsphase unterstützen. Notwendig ist auch die strukturierte Archivierung von Messprogrammen und Messergebnissen. Oft ist es notwendig, auf durchgeführte Messungen zurückzugreifen (z. B. Wiederholungsmessung) oder Messdaten im Sinne der Qualitätssicherung zu speichern.

- Teleservice

  Oft kann ein direktes Bedienen der Maschine nicht unmittelbar an der NPM-Maschine und Bedien-PC erfolgen. Dann müssen entsprechende Optionen vorhanden sein, die die Bedienung der Maschine über ein Netzwerk, im Extremfall über das Internet, hinweg ermöglichen. Diese Funktionen sind besonders für die Ferndiagnose von Fehlerfällen notwendig. Einen wichtigen Punkt im Zusammenhang mit Teleservice stellen Sicherheitsfragen dar. Zum einen muss sichergestellt werden, dass die Fernsteuerung der Maschine nur autorisierten Nutzern vorbehalten bleibt. Zum Zweiten müssen geeignete Mechanismen den Ausfall der Verbindung zwischen der Maschine und einem entfernten Bediener behandeln können. Und zum Dritten muss sichergestellt sein, dass die Übertragung von

Messdaten nicht durch Dritte abgefangen werden kann. Denn dadurch könnten sensible Geometriedaten von Bauteilen in fremde Hände geraten.

Erweitert werden diese Grundfunktionen von Programmteilen, die für den Anwender im Verborgenen laufen. Für die Abarbeitung der erstellten Messprogramme ist beispielsweise ein Interpreter notwendig, der die Messprogramme in für die Maschine verständliche Anweisungen übersetzt. Aber auch die Verwaltung durchgeführter Messvorgänge, um beispielsweise durchgeführte Messungen zu archivieren, muss in ein durchgängiges Steuerungskonzept integriert werden. In Bild 11 sind die Komponenten schematisch in fünf Gruppen eingeteilt dargestellt.

Im Rahmen dieser Dissertation sind Programmteile aus den Bereichen Konfiguration (Diagnosemodul, Kapitel 6.3), grafischer Programmeditor (Messprogrammeditor, Kapitel 6.2), Teleservice (Diagnosemodul, Kapitel 6.3 und Handsteuerung, Kapitel 7.2) und Abarbeitung von Messprogrammen (Messprogramminterpreter, Kapitel 6.4 und Echtzeitvisualisierung, Kapitel 7.2) erstellt worden. Im Zusammenhang mit den Arbeiten am SFB 622 sind im Fachgebiet „Rechneranwendung im Maschinenbau“ weitere Programme entstanden, die in Kapitel 8 beschrieben werden. Dazu zählen unter anderem ein Datenbankprogramm zur Verwaltung von Messergebnissen und ein Visualisierungstool für Messergebnisdateien.

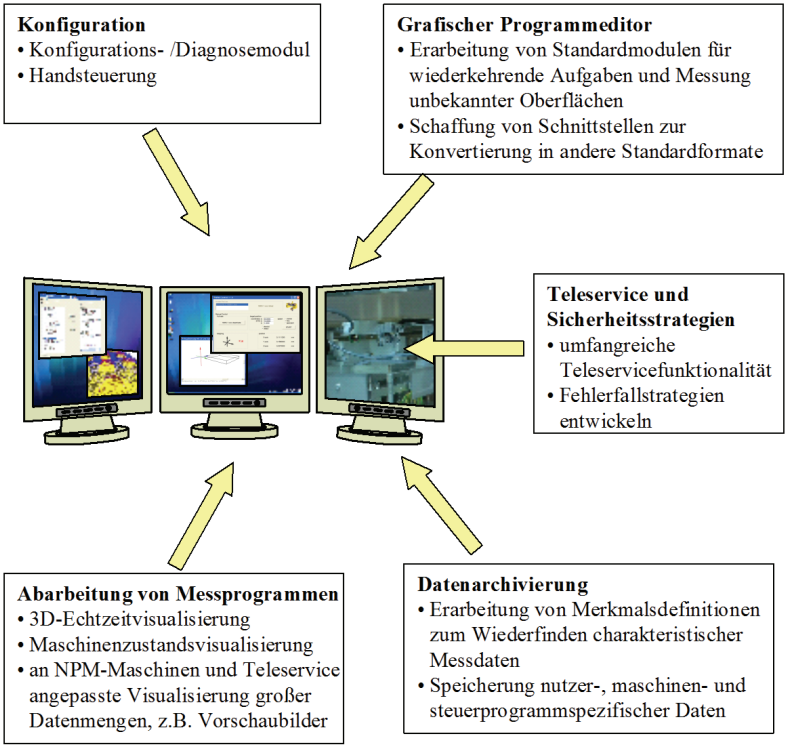


Bild 11: Komponenten einer modularen Steuerung für NPM-Maschinen

3.1 Messprogrammeditor und Programmabarbeitung

Das wichtigste Aufgabengebiet ist im Bereich der Programmerstellung für NPM-Maschinen zu sehen. Unter *Programmen für NPM-Maschinen* sind Dateien zu verstehen, die durch Auflistung bestimmter Maschinenbefehle eine konkrete Aufgabe realisieren. Demnach beinhalten Programme zum Vermessen von Oberflächen Befehle zur Bewegung des Positioniertisches, Befehle zur Antastung mit einem Messsystem und Befehle zur Bewegungssteuerung (z. B. Verfahrgeschwindigkeit). Andere Programme wiederum beinhalten keine Messbefehle, sondern beispielsweise lediglich Positionierbefehle oder Konfigurationsdaten. Trotzdem soll im Zusammenhang mit der Erstellung von Programmen für NPM-Maschinen der Begriff Messprogramme als Bezeichnung für die gesamte Gruppe von Programmen eingeführt werden.

Um den vollständigen Befehlsumfang der Maschine für Programmerstellung nutzen zu können, muss dieser dem Nutzer der Maschine in geeigneter Art und Weise angeboten

werden. In Kapitel 2.2.3 wurde bereits eine Möglichkeit, die Verwendung einer Skriptsprache, dargestellt. Vorteile einer Skriptsprache ist der begrenzte, speziell für die Aufgabe angepasste Syntaxumfang. Nachteilig ist allerdings die schlechte Erweiterbarkeit bei einer Überarbeitung des Maschinenbefehlssatzes. Zudem muss die Skriptsprache wie eine Programmiersprache erlernt werden und ist damit eher für Programmierer geeignet als für die Gruppe von Anwendern, die lediglich spezielle Messaufgaben durchführen wollen. Sinnvoller ist daher der Ansatz, den Maschinenbefehlssatz auf grafische Art und Weise nutzbar zu machen. In [26] wird der wachsende Anspruch an die Software im Bereich der Mess- und Automatisierungssoftware hervorgehoben:

> „Wurde [...] der Einsatz von Windows für Mess- und Automatisierungsanwendungen noch belächelt, so spielen heute Werkzeuge für die grafische Applikationsentwicklung gerade unter Windows mittlerweile in allen Anwendungsbereichen eine immer wichtigere Rolle."

Die Palette reicht dabei von blockorientierten Anwendungen bis hin zu vollständigen grafischen Programmiersprachen (wie z. B. LabVIEW).

Für die Aufgabenstellung bedeutet die Entwicklung eines grafischen Programmeditors, dass zunächst der verfügbare Befehlssatz der Maschine so aufbereitet wird, dass er sich nahtlos in den Editor integrieren lässt. Besonderes Augenmerk ist dabei auf die Erweiterbarkeit des Befehlsumfanges zu legen. Erweiterungen sollen schnell und ohne Änderung in der Programmierumgebung erfolgen können. In der Dissertation wird deshalb der Vorschlag verwirklicht, den Befehlssatz in einer XML-Datei abzubilden. Die konkreten Begründungen und Vorteile dieses Dateiformates werden an entsprechender Stelle in Kapitel 5 erläutert.

Die Abarbeitung von Messprogrammen wird durch einen Interpreter realisiert. Dieser ist in einer ersten Version in das Diagnosemodul eingearbeitet worden. In den Kapiteln 6.3 und 6.4 werden die dabei erzielten Ergebnisse beschrieben.

3.2 Teleservice und Diagnosemodul

Ein direktes Arbeiten mit der NPM-Maschine vom Bedien-PC aus ist in vielen Situationen nicht möglich. So haben beispielsweise Untersuchungen gezeigt, dass bereits die Körperwärme eines im Raum befindlichen Mitarbeiters die Messergebnisse signifikant beeinflusst. Die Verbindung zwischen NPM-Maschine und Bedien-PC, das USB-Kabel, ist aus technologischen Gründen nicht beliebig verlängerbar. Eine Aufstellung des Bedien-PCs in einem Nachbarraum ist also schwierig. Konkrete Gegenmaßnahmen sind für diesen Fall durch einen klimatisierten, abgeschlossenen Arbeitsraum getroffen worden. Allerdings sind

nicht alle Situationen durch Zusatzeinrichtungen wie Abschirmungen zu entschärfen. Dazu zählen solche Fälle, in denen die Bedienung der Maschine aus der Ferne erfolgen soll. Denkbar ist beispielsweise die Überwachung eines langwierigen Messvorganges aus anderen Gebäudeteilen heraus. Auch die Überwachung, Kalibrierung und Diagnose einer Maschine, die sich in einem anderen Erdteil befindet, ist notwendig, um beispielsweise aufgetretene Fehler bereits aus der Ferne lokalisieren und im Idealfall direkt beseitigen zu können. All diese Fälle zeigen, dass die Anforderungen an eine Steuerung von NPM-Maschinen auch Funktionen zur Fernsteuerung einschließen müssen!

Die Aufgabe, ein Steuerungskonzept für NPM-Maschinen zu entwickeln, muss sich also auch diesem Themengebiet widmen. Es muss untersucht werden, welche Technologien sich für die Realisierung entsprechender Aufgaben eignen und wie sich diese modular in das Konzept integrieren lassen. Die Ergebnisse dieser Untersuchungen werden in den Abschnitten des Kapitels 7 beschrieben. Es wird anhand zweier Programmmodule, dem Diagnosemodul (Kapitel 6.3) und einer Handsteuerung (Kapitel 7.2) gezeigt, dass die Realisierung von Teleservicefunktionen für NPM-Maschinen möglich ist. Im Rahmen der Programmierung der Handsteuerung wurden bereits erste Ergebnisse in der dreidimensionalen Zustandsvisualisierung eingearbeitet. Dazu wurde ein Programm entwickelt, welches in einem zusätzlichen Fenster die Position des Messtisches in Echtzeit darstellt.

4 Stand der Technik

Die Nanopositionier- und Nanomessmaschine ist ein universelles Werkzeug zur Positionierung und Vermessung von Objekten im Nanometerbereich. In den vorangegangenen Kapiteln wurde deutlich, dass eine NPM-Maschine aus einer Vielzahl mechanischer, optoelektronischer und elektronischer Komponenten besteht. Die Steuerung der Maschine erfolgt über entsprechende Software, welche auf einem an die Maschine angeschlossenen Bedien-PC läuft.

Im Laufe der Entwicklung eines ersten Prototyps der NPM-Maschine an der Technischen Universität Ilmenau sind spezielle Softwarekomponenten für diese Maschine entstanden. Dazu zählen das Controller-Programm der DSP-Einheit der Maschine mit den darin implementierten Maschinenfunktionen (Firmware), der USB-Treiber für die Kommunikation zwischen der Maschine und dem Bedien-PC und eine Skriptsprache zur Anwendung der Maschinenbefehle.

Das Ziel der Dissertation ist es, eine modulare Steuerung für NPM-Maschinen zu entwerfen. Dafür mussten einerseits die Defizite der bestehenden Softwarekomponenten untersucht werden und andererseits, auf der Basis der modernsten Softwaretechnologien, eine neue Struktur für die modulare Steuerung entworfen, programmiert und implementiert werden. Um dabei den Stand der Technik zu berücksichtigen, ist dieser zunächst zu analysieren. Die folgenden Abschnitte sollen einen kurzen Überblick über den Stand der Technik im Bereich der Steuerung von Werkzeug- und Koordinatenmessmaschinen geben. Zusätzlich werden auch die Programme zur Fernsteuerung (Teleservice) von PCs berücksichtigt.

4.1 Übersicht über ausgewählte Werkzeugmaschinensteuerungen

Eine der vielfältigen Anwendungsmöglichkeiten von NPM-Maschinen ist die Positionierung von Messobjekten, um diese beispielsweise einer Manipulation durch Nano-Tools zuzuführen. Da diese Vorgänge starke Parallelen zum Einsatz von Werkzeugmaschinen aufweisen, soll der Stand der Technik auf diesem Gebiet untersucht werden.

Bei der Steuerung von Werkzeugmaschinen ist vorwiegend die Programmierung nach DIN 66025 anzutreffen. Dafür haben sich in den letzten Jahren viele verschiedene Systeme zur Erstellung der Datensätze etabliert. In Bild 12 ist der Ablauf bei der Programmierung von Werkzeugmaschinen dargestellt. Daneben wird analog dazu der Ablauf für die Programmierung von Spezialmaschinen, dazu zählt beispielsweise auch die NPM-Maschine, gezeigt. Der NC-Interpreter übersetzt die NC-Daten in entsprechende Steuerbefehle für die jeweilige Maschine. Es wird dabei auch von einem sogenannten Postprozessor gesprochen.

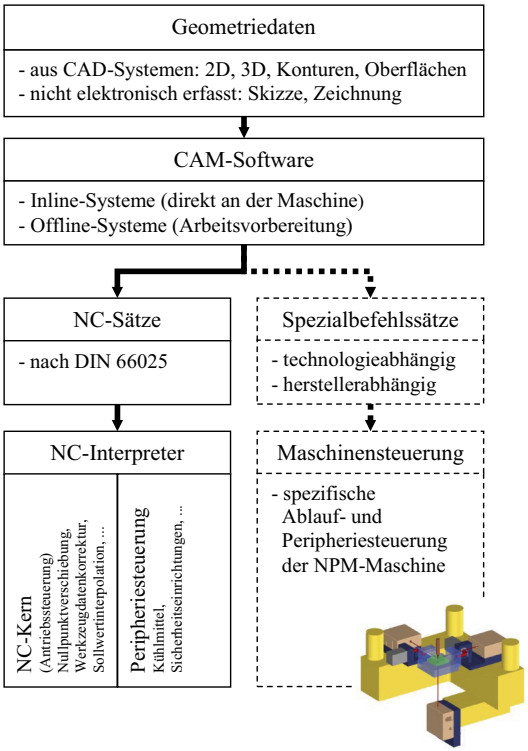


Bild 12: Programmierung von Werkzeugmaschinen

Die Erstellung von NC-Sätzen nach DIN 66025 ist sowohl online, d. h. direkt an der Maschine, als auch offline, z. B. an einem Arbeitsplatz-PC in der Arbeitsvorbereitung, möglich. Die Leistungsfähigkeit der dafür eingesetzten Programme unterscheidet sich stark, in Abhängigkeit von Anwendungsfall und Hersteller.

Die Offline-Programmiersysteme bieten in der Regel ein durchgängiges Programmierkonzept an. Dazu zählen:

- Zeichnungsimport aus unterschiedlichen CAD-Systemen
- Geometriemodul zur Erstellung beliebiger Konturen
- Visuelle Simulation des Arbeitsvorganges und Berechnung der Fertigungszeit
- Generierung der NC-Sätze
- Projektverwaltung
- Datenübertragung an die Maschine

Aber auch die Online-Systeme bieten dem Maschinenbediener leistungsfähige Werkzeuge zur Unterstützung der Programmerstellung an. In der Regel werden in Online-Systemen spezifische Parameter für die Peripheriesteuerung eingestellt oder Fehler in bestehenden NC-Sätzen korrigiert.

Im Folgenden werden beispielhaft für das Gebiet der NC-Programmierung wichtige Softwarelösungen angegeben. Bild 13 zeigt eine Zusammenstellung mehrerer Bildschirm-Snap-Shots des Programms EXSL der Firma SL Automatisierungstechnik.

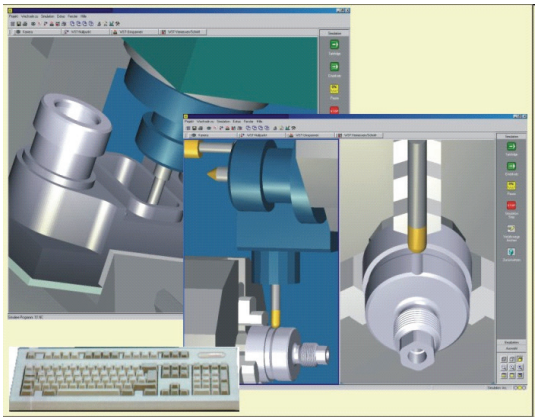


Bild 13: Programmfenster der Software EXSL, Quelle: SL Automatisierungstechnik

Offline-Systeme

- Externes Programmiersystem von SL für Windows (EXSL WIN) [48]
  Ein Programmiersystem der Firma SL Automatisierungstechnik GmbH, Iserlohn. Es existieren Varianten für das Drehen (2D) und für das Fräsen (3D).

- TOPCAM, TOPTURN, TOPMILL [32]
  Durchgängiges Programmiersystem der Firma Mathematisch Technische Software-Entwicklung GmbH, Berlin. Bietet ebenfalls Programmteile für 2D-Anwendungen (TOPTURN) und 3D-Anwendungen an (TOPMILL).

- hyperMILL [38]
  Programmiersystem der Firma Open Mind, Wessling. Die Besonderheit hieran ist die Möglichkeit zur Integration in SolidWorks, ProENGINEER oder Autodesk Inventor.

- PowerMILL [11]
  Mehrachssystem der englischen Firma Delcam.

- NX4 [44]
  NX4 ist ein Teil eines umfangreichen CAD/CAM-Systems der Siemens-Tochter Unigraphics (UGS).

- edgeCAM [7]
  CAM-System der Firma CAMTECH aus Remscheid.

Online-Systeme

Führend im Bereich der Entwicklung von Online-Systemen zur Maschinensteuerung ist die Firma Heidenhain. Sie bietet unter anderem Bahnsteuerungen für Dreh- und Fräsmaschinen an. Mithilfe dieser Systeme ist die direkte Steuerung von Antrieben möglich. Bild 14 zeigt das Terminal einer Steuerung für Drehmaschinen. Ein Touchpanel und weitere Bedienelemente erlauben so Editieren von Steuerprogrammen.

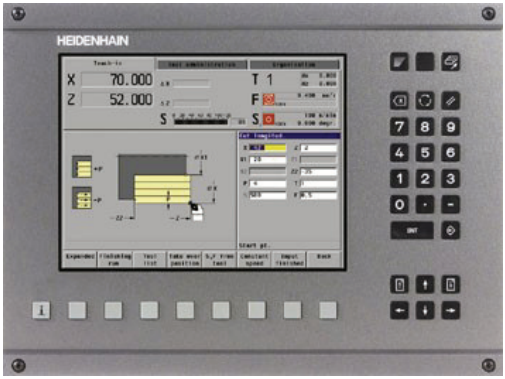


Bild 14: Bahnsteuerung von Drehmaschinen, MANUALplus 4110, Quelle: Heidenhain GmbH

Zusammenfassend lässt sich sagen, dass alle angesprochenen Systeme dazu dienen, Datensätze zur Steuerung von Werkzeugmaschinen zu generieren oder zu editieren. Für den Einsatz zur Steuerung von NPM-Maschinen sind diese Systeme allerdings nicht ohne Weiteres nutzbar. Zum einen existieren keine Postprozessoren, die eine Übersetzung in den Maschinenbefehlssatz der NPM-Maschine erlauben, zum anderen ist der Funktionsumfang und das Aufgabengebiet der NPM-Maschine weitaus größer, als dies bei Werkzeugmaschinen der Fall ist. Für Spezialanwendungen sollte in Zukunft allerdings auch die Entwicklung eines speziellen Konverters von NC-Sätzen in NPM-Befehle in Angriff genommen werden.

4.2 Steuerung von Koordinatenmessmaschinen

Ein weiteres wichtiges Einsatzgebiet von NPM-Maschinen ist das Vermessen von Objekten. Für Zwecke der Qualitätssicherung ist es erforderlich die Maßhaltigkeit bestimmter Strukturen nachzuweisen. In der technischen Praxis existieren für diese Aufgabe Koordinatenmessmaschinen unterschiedlichster Bauart und Baugröße. Im Nanometerbereich soll die NPM-Maschine diese Messaufgaben durchführen. Aus diesem Grund ist zu prüfen, ob es Parallelen zu Softwareprodukten aus der Messtechnik gibt.

Ein führendes Unternehmen im Bereich Messtechnik ist die Carl Zeiss AG. Sie bietet neben Messgeräten auch eine Vielzahl von Softwarelösungen an, um unterschiedlichste Messaufgaben zu realisieren. Darunter sind beispielsweise Programme zur Rekonstruktion von Freiformflächen, zur Vermessung spezieller Geometrien oder zur Generierung von Messprogrammen aus CAD-Daten. Aber auch andere Hersteller liefern entsprechende Software. Einige Beispiele für Messsoftware sind:

- Carl Zeiss AG
  - Calypso: Software zur Erstellung von Messprogrammen für Zeiss-Geräte direkt aus CAD-Daten.
  - CMM-OS: Programmierschnittstelle zur Integration der Funktionen von Zeiss-Geräten in eigene Applikationen.

- Hexagon Metrology GmbH (ehemals Leitz Messtechnik GmbH)
  - QUINDOS: Modulares Softwarepaket zur Realisierung unterschiedlichster Messaufgaben für Koordinatenmessgeräte.
  - PC-DMIS: Messsoftware für Hexagon-Geräte. Auch hier ist die Generierung von Messprogrammen aus CAD-Daten möglich.

- Werth Messtechnik GmbH
  - WinWerth: 3D-Messsoftware

Neben diesen wenigen Beispielen existieren viele weitere Softwarepakete anderer Hersteller. Viele Programme bieten eine Schnittstelle, um CAD-Daten als Basis für die Erstellung von Messprogrammen nutzen zu können. Allerdings sind diese Messprogramme nur in den seltensten Fällen auf Messmaschinen eines anderen Herstellers übertragbar. Positives Gegenbeispiel ist das Softwarepaket Calypso, bei dem Messprogramme auf verschiedenen Maschinen abgearbeitet werden können [18].

Allerdings gibt es Bestrebungen, ein einheitliches Datenaustauschformat zwischen verschiedenen Softwarelösungen und Messgeräten zu etablieren. Durch die *International Association of CMM Manufacturers* (IA.CMM; CMM – Coordinate Measuring Machine, Koordinatenmessgerät) ist dazu, unter Anregung namhafter Automobilhersteller, die Schnittstelle I++/DME (Interface Dimensional Measuring Equipment) geschaffen worden. Diese standardisierte Schnittstelle soll es ermöglichen, dass Messprogramme einer speziellen Software auf verschiedenen Messgeräten ausgeführt werden können. Dies hat für die Automobilhersteller den Vorteil, dass sie nicht mehr an eine bestimmte Messprogramm-Messgerät-Kombination gebunden sind. Sobald sowohl Messsoftware, als auch Messgerät die I++/DME-Schnittstelle implementieren, ist eine Kommunikation unterschiedlicher Systeme möglich. Zur Verdeutlichung dient Bild 15.

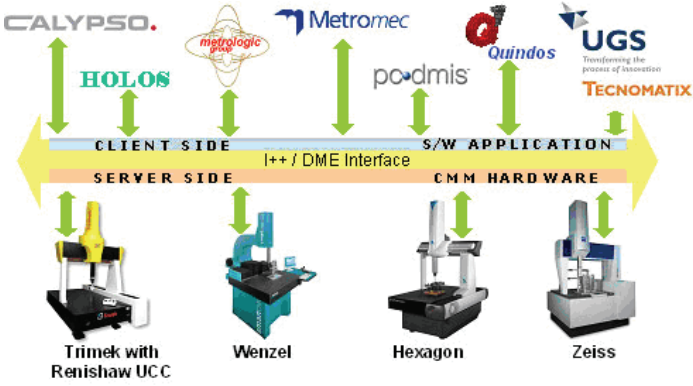


Bild 15: Schematische Darstellung der I++/DME-Schnittstelle, Quelle: www.iacmm.com

Im Rahmen des Sonderforschungsbereiches 622 an der Technischen Universität Ilmenau ist im Fachgebiet Qualitätssicherung eine erste Anbindung der NPM-Maschine an einen

I++/DME-Server erfolgt. Eine Ansteuerung der NPM-Maschine, beispielsweise auch mit Calypso, ist damit möglich.

Die Anwendungsmöglichkeiten der NPM-Maschine reichen aber über den Bereich der Koordinatenmessung hinaus. Im Rahmen der Entwicklung einer modularen Steuerung für NPM-Maschinen müssen neben der Berücksichtigung des angesprochenen Standards, weitere Komponenten entwickelt und in das Gesamtkonzept integriert werden.

4.3 Systeme zur Realisierung von Teleservice-Funktionalitäten

Es gibt eine Reihe von kommerziellen Programmen, die es ermöglichen einen entfernten Rechner fernzusteuern. Im Allgemeinen spricht man von Remote Control Software. In Windows XP ist ein solches Programm bereits integriert. Es heißt Remote Desktop Connection. Weitere Beispiele sind:

- Remote Assistance
  Basiert auf Remote Desktop Connection und ist ebenfalls in Windows integriert. Zusätzlich zu Remote Desktop Connection bietet Remote Assistance ein Chat-Fenster und die Möglichkeit eine Ton- und Videoverbindung zu schalten.

- PCAnywhere von Symantec

- VNC (Virtual Network Computing)
  Ein Open-Source-Projekt von AT&T Labs Cambridge.

Je nach Anwendungsfall unterscheiden sich auch die Technologien, auf denen die Programme basieren. Ein Anwendungsszenarium ist die Überwachung von Rechnern über große Entfernungen hinweg. Dabei werden in der Regel lediglich der Bildschirminhalt des zu steuernden Rechners und die Tastatur- und Mauseingaben des steuernden Rechners ausgetauscht. Ein anderer Anwendungsfall ist im Einsatz sogenannter *Thin Clients* zu sehen. Hierbei läuft ein ressourcenintensives Programm auf einem leistungsfähigen Computer. Ein „kleinerer“ Computer dient lediglich als Eingabe- und Ausgabe-Terminal des Servers.

In allen Fällen muss aber sowohl auf dem Server als auch auf dem Client spezielle Software installiert werden. Die Konfiguration der Software auf dem Server und auf dem Client ist aufwendig. Zum Teil sind umfangreiche Änderungen an eventuell vorhandenen Firewalls vorzunehmen.

Für den Einsatz im Rahmen einer modularen Steuerung für NPM-Maschinen kommen existierende Remote Control Programme aus folgenden Gründen nicht zum Einsatz:

- Für die Teleservicefunktionen für die NPM-Maschine ist es nicht erforderlich den Bildschirminhalt des Servers auf dem Client darzustellen. Es genügt lediglich der Aufruf spezieller Maschinenfunktionen über das Netzwerk hinweg. Die Rückgabewerte werden dann in einem entsprechenden Client-Programm dargestellt. Bestehende Programme können diese Funktionsaufrufe nicht gewährleisten.

- Kommerzielle Programme bieten in der Regel keine Möglichkeit diese in das Steuerungskonzept der NPM-Maschine einzubinden.

Für die Realisierung der Teleservicefunktionen für die NPM-Maschine sollen deshalb eigene Programme geschaffen werden. Nur so kann sichergestellt werden, dass die Kommunikation zwischen Client und Server, beispielsweise zwischen einem Diagnosemodul und der NPM-Maschine, die hohen Sicherheitsanforderungen erfüllt. Für die Realisierung existiert eine Reihe möglicher Technologien. Diese werden in Kapitel 7 erläutert.

5 XML als Basis eines gemeinsamen Datenformates

Kaum ein anderes Datenformat hat in den letzten Jahren eine derartige Verbreitung erlebt wie das Datenformat XML (*Extensible Markup Language*). Die Gründe für diese Entwicklung liegen zum einen in den Eigenschaften von XML selbst und zum anderen in einer *Quasi-Standardisierung* durch das *World Wide Web Consortium* (W3C). Das W3C ist eine Vereinigung verschiedenster Organisationen, Firmen und Privatpersonen. Sie hat sich zum Ziel gestellt einheitliche Standards für das Internet zu entwickeln. Obwohl diese Standards keinen normativen Charakter besitzen, sind sie in der Regel allgemein anerkannt und werden entsprechend verwendet. Einige wichtige Empfehlungen des W3Cs sind, neben XML, beispielsweise auch *Hypertext Markup Language* (HTML) und die *Cascading Style Sheets* (CSS).

Um die Vorteile und den Nutzen, den das Datenformat XML mit sich bringt zu verdeutlichen, werden im Folgenden zunächst die Grundlagen von XML erläutert. Die Erläuterungen sollen sich allerdings nur auf die allerwichtigsten Grundlagen beziehen. Weitere Informationen sind beispielsweise in [17] und [60] zu finden.

5.1 Grundlagen von XML

XML ist eng mit der Entwicklung von Textverarbeitungssystemen verknüpft [17]. Deshalb lässt sich die Idee von XML hieran besonders leicht verdeutlichen. Texte bestehen aus verschiedenen Komponenten wie Überschriften, Absätzen oder Wörtern. Der Inhalt der Komponenten sagt zunächst nichts über dessen Zuordnung zu einer der genannten Komponenten aus. Es muss daher ein System gefunden werden, den Inhalt so zu kennzeichnen, dass sofort klar wird, ob es sich um eine Überschrift, einen Absatz oder um eine Aufzählung handelt. Diese Kennzeichnung wird auch Textauszeichnung, bzw. im Englischen *Markup*, genannt. Frühzeitig erkannte man, dass eine Vereinheitlichung dieser Auszeichnungen für die Beschreibung von Textdokumenten getroffen werden musste, um den Austausch von Dokumenten zu ermöglichen. Bereits in den 1960er Jahren versuchte Charles Goldfarb die verschiedenen Textverarbeitungssysteme der Firma IBM zusammenzuführen.

Die erste Lösung war die *Document Composing Facility Generalized Markup Language* (DCF GML). Sie sollte erstmals den Inhalt von Textdokumenten unabhängig von der verwendeten Hard- und Software beschreiben. Die Weiterentwicklung dieses Ansatzes führte im Jahr 1986 zu einer durch die ISO (International Organisation for Standardisation) normierten Auszeichnungssprache: ISO 8879: *Information Processing — Text and Office Systems — Standard Generalized Markup Language* (SGML) [22]. SGML sollte selbst dazu dienen, neue Auszeichnungssprachen zu definieren und beinhaltet daher eine nahezu unüberschaubare Anzahl an Vorschriften und Regeln. Eine konkrete Anwendung von SGML war die 1989 eingeführte Sprache HTML. Diese diente der Beschreibung von Internetseiten. Der Umfang von HTML wurde aber schnell zu klein, sodass verschiedene, zum Teil firmenspezifische (proprietäre) Erweiterungen vorgenommen wurden. Die Erweiterungen konnten aber immer nur kurzfristig den rasant wachsenden Anspruch an die Darstellung von Internetseiten gerecht werden. Gleichzeitig wuchs mit zunehmender Dynamisierung der Inhalte von Webseiten das Problem der Trennung von Inhalt und Darstellung. Durch Verwendung von SGML konnten diese Nachteile vermieden werden. Allerdings ist die Komplexität von SGML so hoch, dass die Verwendung des gesamten Regelwerkes nicht möglich ist. Deshalb wurde auf Basis von SGML eine neue Auszeichnungssprache definiert, die nur eine Teilmenge von SGML beinhaltet: XML. Am 10. Februar 1998 verabschiedete das W3C die Empfehlung für XML 1.0. Diese Empfehlung ist bis heute gültig.

Technisch gesehen sind XML-Daten ASCII-Zeichen, gespeichert als Dateien oder als Bytefolge im Arbeitsspeicher eines Prozesses. Die logische Struktur von XML basiert auf *Elementen*. Alle in einem XML-Dokument gespeicherten Informationen werden durch die Zuordnung zu Elementen in eine logische Struktur gebracht. Die einzelnen Elemente sind durch einen Namen gekennzeichnet. Die Namen sind frei wählbar, müssen aber mit einem Buchstaben oder einem Unterstrich beginnen. Ein Element beginnt mit einem *öffnenden Tag* <*ElementName*> und endet mit einem *schließenden Tag* </*ElementName*>. Den Inhalt von Elementen können Werte oder auch weitere Elemente bilden. Zusätzlich darf ein Element leer sein: <*ElementName* />. Im unten aufgeführten Beispiel ist ein vollständiges XML-Dokument zu sehen.

Die Abbildung der Daten innerhalb einer XML-Struktur kann beliebig komplex sein. Die daraus resultierende Struktur eines XML-Dokuments kann auch als Baum betrachtet werden. Ausgehend von der Wurzel, repräsentiert von dem unbedingt notwendigen *Wurzelelement*, können sich die weiteren Elemente beliebig verzweigen. Die Elemente können beliebig

ineinander geschachtelt werden. Voraussetzung ist, dass die Schachtelung symmetrisch erfolgt, d. h., öffnende und schließende Tags von Elementen dürfen sich nicht überschneiden.

Eine konkrete Datei könnte folgenden Inhalt haben:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<tui:Adressen xmlns:tui="www.TU-Ilmenau.de/ram">
  <tui:Person>
    <tui:Name>Schmidt</tui:Name>
    <tui:Vorname>Peter</tui:Vorname>
    <tui:Anschrift art="privat">
      <tui:Strasse>Waldweg</tui:Strasse>
      <tui:Hausnummer>14</tui:Hausnummer>
      <tui:Ort>Waldstadt</tui:Ort>
      <tui:PLZ>12345</tui:PLZ>
    </tui:Anschrift>
    <tui:Telefon art="privat">
      <tui:Festnetz>03655-777777</tui:Festnetz>
      <tui:Mobil>0176-12345678</tui:Mobil>
    </tui:Telefon>
    <tui:Telefon art="dienstlich">
      <tui:Festnetz>03655-789654</tui:Festnetz>
    </tui:Telefon>
  </tui:Person>
</tui:Adressen>
```

Durch die freie Wahl der Elementnamen können Überschneidungen in der Bedeutung einiger Elemente auftreten. Beispielsweise kann eine XML-Datei, die den Bestand an Hardware einer Firma speichert, den Elementname „Name“ sowohl für die Bezeichnung eines Rechners, als auch für die Angabe einer verantwortlichen Person verwenden. Um derartige Mehrdeutigkeiten auszuschließen, wurden sogenannte *Namespaces* (Namensräume) eingeführt. Namensräume sind eindeutige Zeichenketten, die dem Elementnamen als Präfix zugeordnet werden können. Um die Eindeutigkeit zu gewährleisten, hat es sich als sinnvoll erwiesen, Internetadressen als Namensräume zu verwenden. Anhand dieser Präfixe ist die Unterscheidung namensgleicher Elemente dennoch möglich. Im obigen Beispiel wurde ein Namensraum *www.TU-Ilmenau.de/ram* mit dem dazugehörigen Präfix *tui* eingeführt.

Elemente können durch *Attribute* näher beschrieben werden. In der abgedruckten Adressdatei gibt es beispielsweise das Attribut *art*. Es beschreibt den Kontext der gespeicherten Daten hinsichtlich privater oder dienstlicher Kontakte näher. Die Regeln zur Namensbildung von

Attributen sind mit der von Elementnamen identisch. Der Wert von Attributen wird in Anführungszeichen angeben.

Ein weiteres wichtiges Merkmal von XML-Dokumenten ist die *XML-Deklaration*. Diese wird als erste Zeile der XML-Datei eingefügt und stellt eine spezielle Form von Verarbeitungsanweisung dar. Sie gibt die Version (derzeit nur 1.0) und die Codierung des Zeichensatzes an. Weitere Verarbeitungsweisungen können die Angabe von Stylesheets und Schemata sein.

Entspricht eine Datei den Regeln des XML-Standards, ist also die Deklaration vorhanden, existiert genau ein Wurzelelement und sind alle Elemente korrekt ineinander verschachtelt, so spricht man von einem *wohlgeformten Dokument*. Die Wohlgeformtheit gibt allerdings nur Aufschluss über die Einhaltung des XML-Standards. Möchte man sicherstellen, dass ein Dokument einer speziellen Struktur entspricht, also einem speziellen Muster entspricht, muss ein Schema definiert werden (siehe Kapitel 5.2.2). Solche Schemata können beispielsweise festlegen, wie die Elemente zu heißen haben, welche Reihenfolge und Schachtelung die Elemente aufweisen müssen oder welchen Wertebereich die Inhalte von Elementen und Attributen umfassen dürfen. Entspricht ein Dokument diesen Forderungen, so handelt es sich um ein *gültiges Dokument*. Beim Verarbeiten eines gültigen Dokumentes ist sichergestellt, dass alle notwendige Informationen enthalten sind.

Das Erstellen von XML-Dateien kann auf einfachste Art und Weise mit jedem Texteditor erfolgen. Es gibt aber eine unüberschaubare Zahl an speziellen XML-Editoren. Beispielsweise bietet das Visual Studio von Microsoft einen komfortablen XML-Editor.

Allein die Tatsache, dass es sich bei XML um ein textbasiertes Datenformat handelt, bei dem beliebige Daten durch frei wählbare Auszeichnungselemente strukturiert werden können, stellt noch keinen Vorteil gegenüber anderen Datenformaten dar. Erst ein Blick auf einen Teil der gesamten Familie an Empfehlungen und Standards im Zusammenhang mit XML zeigt, dass es sich bei XML um ein Datenformat mit sehr vielen Vorteilen handelt.

5.2 Einige Technologien der XML-Familie

5.2.1 XML-Parser

Als *XML-Prozessoren* bezeichnet man Programme, die in der Lage sind, XML-Dokumente zu verarbeiten und den Zugriff auf deren Inhalt ermöglichen. Ein wichtiger Teil solcher Programme sind sogenannte *Parser*. Der Parser analysiert die Auszeichnungen des Dokumentes und ist somit in der Lage die Struktur zu bestimmen. Ein Parser zerlegt also ein

XML-Dokument in seine einzelnen logischen Komponenten. Dabei kann er auf die Regeln zurückgreifen, die durch die XML-Definition festgelegt sind [60].

Parser lassen sich nach verschiedenen Kriterien ordnen. Eine Einteilung kann in *validierende* und *nichtvalidierende* Parser vorgenommen werden. Validierende Parser analysieren die Struktur einer XML-Datei und prüfen, ob diese einem vorgegebenen Schema entspricht. Ein validierender Parser kann also die Gültigkeit einer XML-Datei feststellen. Bei nichtvalidierenden Parsern wird lediglich die Wohlgeformtheit geprüft. Eine Prüfung ob bestimmte Elemente vorhanden sind, kann ein derartiger Parser nicht durchführen. Eine andere Einteilung von Parsern kann anhand der Methode getroffen werden, die dazu verwendet wird, die Komponenten der Struktur bereitzustellen. Unter diesem Gesichtspunkt sind *baumbasierte* und *streambasierte* Parser zu unterscheiden:

Baumbasierte Parser

Wie bereits erwähnt, lässt sich die Struktur eines XML-Dokumentes als Baum darstellen. Ein baumbasierter Parser liest zunächst die gesamte Datei in den Speicher und bildet diese anschließend in einem sogenannten *Knotenbaum* (Bild 16) ab. Als *Knoten* wird ein einzelner Bestandteil der Struktur bezeichnet. Der Zugriff auf die einzelnen Komponenten ist durch die *DOM-API* (Document Object Model – API) gewährleistet. Das DOM bietet verschiedene Schnittstellen, um auf die einzelnen Knoten zuzugreifen. Dazu zählen die Schnittstellen *Node*, *NodeList*, *Element* und *Document* (der Wurzelknoten).

Mithilfe der durch das DOM bereitgestellten Schnittstellen und den darin definierten Methoden und Eigenschaften lassen sich Informationen einer XML_Datei auf sehr einfache Art und Weise ermitteln. Ein baumbasierter Parser arbeitet auf Basis des DOM.

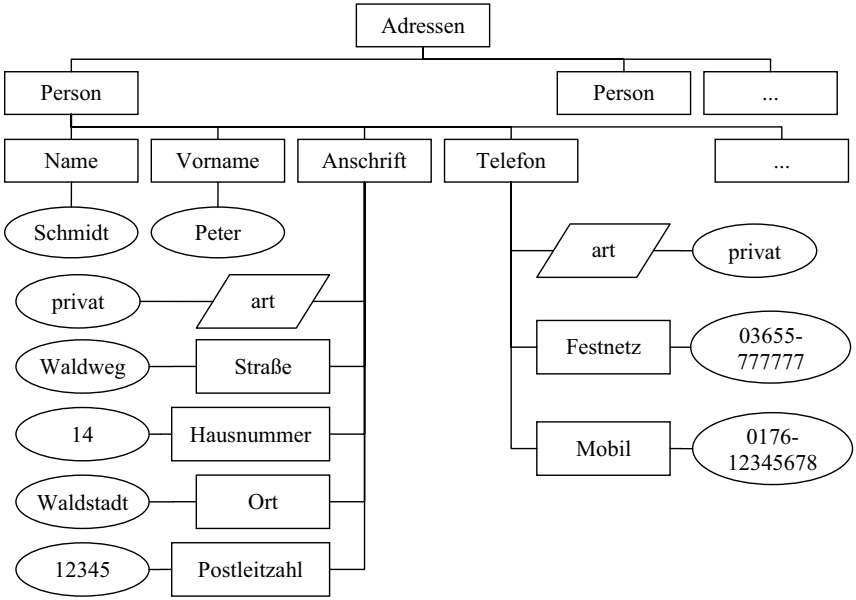


Bild 16: Die Struktur einer XML-Datei nach dem Document Object Model

Streambasierte Parser

Bei der Verwendung des DOM muss zunächst das gesamte Dokument in den Arbeitsspeicher geladen werden. Bei großen Dokumenten ist diese Vorgehensweise sehr ressourcenaufwendig und damit nachteilig. Eine andere Schnittstelle für den Zugriff auf die Komponenten von XML-Dokumenten bietet die *Simple API for XML* (SAX). Hier wird nicht das gesamte Dokument eingelesen. Vielmehr liest der Parser das Dokument partiell und generiert beim Erkennen einer Komponente ein Ereignis. So gibt es Ereignisse für den Beginn und das Ende von Elementen und Attributen sowie für das Erreichen von Element- oder Attributdaten. Mithilfe dieser Ereignisse kann eine Anwendung entsprechende Daten konsumieren.

Streambasierte Parser arbeiten in der Regel nur vorwärtsgerichtet. Das heißt, ein Navigieren ist nicht wie bei baumbasierten Parsen beliebig innerhalb des Dokumentes möglich, sondern nur sequenziell in der Reihenfolge des Auftretens.

XML-Parser können aufgrund der Syntaxregeln der Extensible Markup Language die Struktur von XML-Dokumenten analysieren und den Inhalt in einzelne Komponenten zerlegen. Dabei existiert heute eine ganze Reihe von Parsern die eingesetzt werden können. Im Visual Studio beispielsweise ist eine Vielzahl an Klassen implementiert, die sowohl validierendes, nichtvalidierendes, baumbasiertes als auch streambasiertes parsen von XML-Dokumenten

ermöglichen. Damit ist dem Programmierer ein wichtiges Werkzeug im Umgang mit XML in die Hand gegeben worden. Auf einfache und effiziente Weise können Daten aus einer XML-Datei gelesen werden. Die vorhandenen XML-Parser sichern dadurch dem Datenformat XML einen deutlichen Vorteil gegenüber anderen Datenformaten.

5.2.2 XML-Schema

Die Syntaxregeln der Extensible Markup Language legen nur den strukturellen Aufbau von XML-Dokumenten fest. Nicht festgelegt sind beispielsweise die Namen der Elemente, die Verschachtelung der einzelnen Elemente untereinander oder der Einsatz von Attributen. Damit ist aber die Gefahr verbunden, dass ein Programm, welches seine Daten aus einer XML-Datei bezieht, nicht fehlerfrei ausgeführt werden kann, wenn sich der Aufbau der eingelesenen Datei von dem erwarteten unterscheidet. Fehlen Elemente, sind diese anders benannt oder sind sie anders verschachtelt, können die Daten nicht mehr eingelesen werden. Trotzdem kann es sich bei der Datei um eine wohlgeformte XML-Datei handeln. Dieses Problem kann durch den Einsatz von Schemata behoben werden. Mit Schemata lässt sich der Aufbau eines XML-Dokuments genau festlegen. Schemata können das Vorkommen von Elementen verlangen oder verhindern und die Verschachtelung dieser untereinander vorgeben.

Für XML gibt es zwei verschiedene Arten von Schemata: *Document Type Definition* (DTD) und *XML Schema Definition* (XSD). Beide Varianten kommen heute zum Einsatz, wobei der Schwerpunkt auf der Verwendung von XSD Schemata liegt. Für das Beispiel der obigen Adressdatei sieht das mit dem Visual Studio erzeugte Schemadiagramm wie folgt aus:

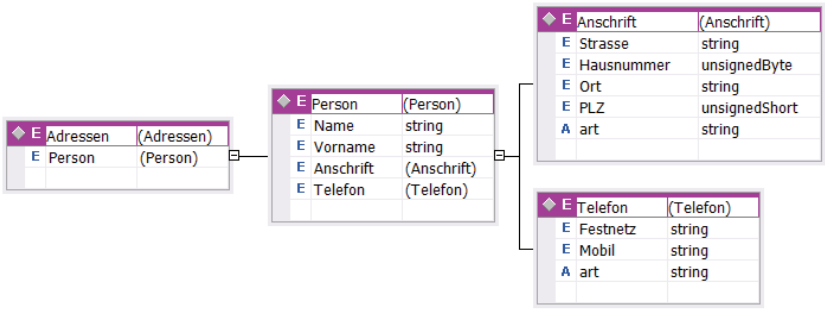


Bild 17: Schema der Adressdatei

Das Diagramm ist eine grafische Darstellung einer dahinterliegenden XML-Datei mit der Endung *.xsd*. In dieser Schemadatei sind alle Bedingungen an eine XML-Datei gespeichert. Einen Auszug aus der Datei stellen folgende Zeilen dar:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema xmlns:tui="www.TU-Ilmenau.de/ram" targetNamespace="www.TU-
              Ilmenau.de/ram" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Adressen">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Person">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Name" type="xs:string" minOccurs="1"
                                                          maxOccurs="1" />
              <xs:element name="Vorname" type="xs:string" minOccurs="1" />
              <xs:element name="Anschrift" minOccurs="1"
                                                  maxOccurs="unbounded">
...
```

Schemata für XML-Dateien bieten also die Möglichkeit, die Struktur der Daten genau festzulegen. Damit ist sichergestellt, dass weiterverarbeitende Programme auf die Daten zugreifen können, die sie benötigen. Schemata stellen damit einen weiteren Vorteil in der Verwendung von XML-Dateien dar.

Die Darstellung der Funktionsweise von Schemata ist an dieser Stelle bewusst kurz gehalten worden. Im Rahmen dieser Dissertation wurden einige Schemata entwickelt, siehe z. B. Kapitel 6.2.6. Allgemeine Informationen zu Schemata sind unter anderem in [47] und [60] zu finden.

5.2.3 Transformationen

Oft kommt es vor, dass das Schema einer XML-Datei nicht mehr den aktuellen Anforderungen der Anwendung entspricht. Um aber bereits bestehende Dateien weiter nutzen zu können, bietet die XML-Familie eine Technologie zur Transformation von XML-Daten an. Die dafür verantwortlichen Techniken heißen *XML Stylesheet Language* (XSL) und *XSL Transformation* (XSLT).

Ein erster Ansatz festzulegen, wie XML-Dokumente formatiert werden sollten, wurde 1999 vom W3C vorgestellt. Dazu wurde XSL-Spezifikation durch Formatierungsobjekte erweitert (*XSL Formatting Objects* – XSL-FO). Damit ist es beispielsweise möglich, XML-Dokumente zu PDF-Dokumenten zu formatieren. An dieser Stelle soll aber nicht weiter auf diese Technik eingegangen werden, da sie nicht Gegenstand der Dissertation war. Nähere Informationen finden sich beispielsweise in [60] auf Seite 132.

Interessanter ist ein weiterer Ableger von XSL: XSLT. Hierdurch kann beispielsweise ein XML-Dokument in eine HTML-Seite transformiert werden. Ebenfalls ist es mit XSLT möglich, XML-Dokumente in andere XML-Dokumente zu transformieren oder mehrere Dokumente zusammenzuführen. Diese Möglichkeiten sind im Rahmen der Dissertation allerdings nur am Rande untersucht worden und sind nicht mittelbarer Bestandteil der Arbeit. Deshalb soll an dieser Stelle nicht weiter darauf eingegangen werden.

XML-Transformationen stellen aber einen weiteren Vorteil von XML dar. Damit können XML-Dokumente schnell in andere Strukturen überführt werden. Dies ist bei anderen Datenformaten oft nur mit spezifischen Konvertern möglich.

5.2.4 Abgeleitete Sprachen

Von XML können weitere Sprachen abgeleitet werden. So ist es in den letzten Jahren verstärkt zur Entwicklung von xml-basierten Sprachen gekommen, um verschiedenste Daten in definierter Art und Weise auszuzeichnen. Um nur einige, wenige Beispiele für diese Entwicklung zu nennen, folgt eine kurze Auflistung:

- Für Texte und Formeln:
  - DocBook — zur Beschreibung von Textdokumenten, siehe www.docbook.de
  - MathML — zur Beschreibung mathematischer Formeln, siehe http://www.w3.org/TR/MathML2/
- Für Grafiken:
  - SVG — Scalable Vector Graphics, Standard zur Beschreibung zweidimensionaler Vektorgrafiken, siehe http://www.w3.org/Graphics/SVG/
  - X3D — 3D-Modellierungssprache, siehe http://web3d.org/
- Für Sicherheit:
  - XML Signature — Spezifikation für digitale Signaturen, siehe http://www.w3.org/Signature/
  - XML Encryption — Spezifikationen zum ver- und entschlüsseln von XML-Dokumenten, siehe http://www.w3.org/Encryption/2001/

Diese Liste ist keinesfalls vollständig oder erhebt den Anspruch wenigstens die bedeutendsten Sprachen zu nennen. Sie soll lediglich verdeutlichen, dass es eine Vielzahl an Sprachen gibt,

die auf der Basis von XML entstanden sind. Für viele Anwendungen hat sich dadurch eine Art Standard zum Speichern der Daten herauskristallisiert. Beispielsweise ist SVG ein weit verbreitetes Datenformat für die Speicherung und die Darstellung von zweidimensionalen Vektorgrafiken im Internet geworden.

5.3 Umsetzung des Maschinenbefehlssatzes in XML

Die vorhergehenden Abschnitte haben gezeigt, dass XML ein weit verbreitetes und zukunftssicheres Datenformat zur Strukturierung unterschiedlichster Daten ist. Darüber hinaus haben sich auf dem Gebiet der Programmierung viele Anwendungen entwickelt, die auf XML zurückgreifen. Beispielsweise werden die Konfigurationsdaten einer Windows-Anwendung, die mithilfe des Visual Studio programmiert wurde, standardmäßig in XML-Dateien abgespeichert. Aber auch andere Daten werden in XML-Dateien serialisiert.

Aus diesem Grund entstand die Überlegung, auch für die Programmierung von Komponenten für die Steuerung von NPM-Maschinen XML als Datenbasis einzusetzen. Ein erster Anwendungsfall bot sich bei der Visualisierung bereits bestehender Messwertdateien an. Die zur Verfügung stehenden Dateien waren in der Regel Textdateien mit unterschiedlichem Aufbau. Um den Inhalt zu visualisieren, musste zunächst manuell geprüft werden, worum es sich bei den einzelnen Werten handelte. Gab es zu der entsprechenden Datei weder Hinweise in schriftlicher Form, noch konnte der Ersteller der Datei Auskunft darüber erteilen, konnten die in den Daten „versteckten“ Messwerte nur mühselig identifiziert werden. Schnell wurde klar, dass die Daten sich selbst beschreiben müssten. Ein klassischer Fall für *Metadaten*. Daraufhin wurde das Datenformat XML eingeführt. Zunächst wurde ein erstes Schema entworfen, um die Daten strukturiert in einer XML-Datei zu speichern. Neben den eigentlichen Daten war es damit auch möglich, zusätzliche Informationen, wie z. B. über den Messablauf oder die Bedeutung der Messwerte, abzuspeichern. Diese Informationen führten zu einer enormen Erleichterung im Umgang mit den Messwertdateien. Mithilfe mehrerer Konverter konnten schließlich alle Messwertdateien in das XML-Format überführt werden. Die Zusatzinformationen konnten nun dem Visualisierungsprogramm Informationen über den Aufbau der Messwertdatei liefern. Statt einer Vielzahl unterschiedlicher Visualisierungsprogramme für alle möglichen Fälle des Aufbaus der Messwertdateien genügte jetzt ein Programm, welches sich den speziellen Bedürfnissen der Datei anpassen konnte.

Die dabei gemachten, guten Erfahrungen ließen den Schluss zu, auch weitere Programmteile auf der Basis von XML-Daten zu erstellen. Als Nächstes sollte ein Programm entwickelt

werden, mit dem es möglich ist Messprogramme für die NPM-Maschine zu schreiben. Die bisherigen Lösungen (siehe Kapitel 2.2.3) waren nur schwer anwendbar. Bereits erstellte Messprogramme hatten das Aussehen von unübersichtlichem Quellcode. Warum sollte es nicht mithilfe von XML-Dateien möglich sein die NPM-Maschine zu steuern? Ein geeignetes Schema könnte dafür sorgen, dass sowohl der Benutzer ein Messprogramm lesen und editieren könnte, als auch die Maschine die benötigten Daten bereitgestellt bekommt. Um später eine durchgängige Anwendung von XML-Daten zu gewährleisten, wurde zunächst der Befehlssatz und daraus resultierend die Messprogrammdateien in einer XML-Struktur abgebildet. Der erstellte Messprogrammeditor konnte nun auf den im XML-Format vorliegenden Befehlssatz der Maschine zugreifen. Die vom Nutzer ausgewählten und parametrierten Befehle wurden anschließend wiederum in einer XML-Datei abgespeichert.

Der Aufbau der XML-Datei, die den Befehlssatz abbildet, ist aus Platzgründen nur teilweise in Bild 18 dargestellt. Das Wurzelelement der Datei heißt *NPM_INSTRUCTION_SET*. Die nächste Ebene wird durch die vier Elemente *INSTRUCTIONS*, *ERROR_VALUES*, *SYMBOLS* und *STRUCTURES* gebildet. Das Instructions-Element bildet den Rahmen für alle weiteren Maschinenfunktionen. Das gesamte Schema einer solchen Funktion ist im Anhang abgebildet. Die anderen Elemente beinhalten die Werte für mögliche Fehlercodes (ERRORS_VALUES), die Werte für definierte Auswahltypen (SYMBOL) und die Definition vorhandener Strukturen (STRUCT).

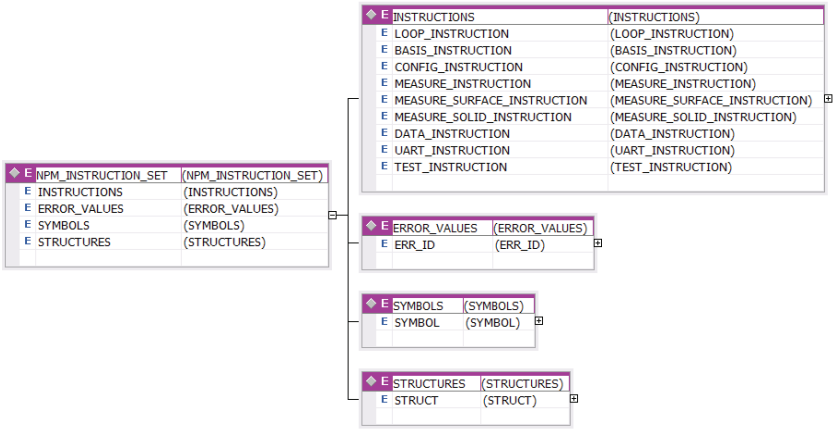


Bild 18: Teil des Schemas der Befehlssatzdatei

Mithilfe dieser Informationen kann der Messprogrammeditor die für die Kommunikation mit der NPM-Maschine notwendigen Bytefelder (siehe Kapitel 2.2) mit Werten füllen und in

Zusammenarbeit mit dem Interpreter entsprechende Funktionsaufrufe durchführen. Wird der Funktionsumfang später innerhalb der DSP-Software erweitert, ist lediglich ein entsprechendes Element in die XML-Befehlsdatei einzufügen. Damit ist es erstmals ermöglicht worden, ohne Änderungen am Messprogrammeditor zusätzliche Maschinenfunktionen in der NPM-Maschine implementieren zu können!

Ein weiterer Vorteil des Befehlssatzes auf der Basis von XML ergibt sich aus der Möglichkeit den Befehlssatz durch geeignete Transformationen an den jeweiligen Nutzertyp anzupassen. Beispielsweise kann der Befehlssatz durch eine Transformation mittels XSLT derart eingeschränkt werden, dass für bestimmte Nutzer nur ein verringerter Funktionsumfang zur Verfügung steht, für Administratoren aber der gesamte Umfang zugänglich ist. Diese Möglichkeit ist aber Gegenstand zukünftiger Arbeiten.

6 Module für die Steuerung von NPM-Maschinen

6.1 Entwicklung eines modularen Gesamtkonzepts einer Steuerung

NPM-Maschinen besitzen, wie bereits erwähnt, aufgrund ihrer technischen Möglichkeiten ein sehr breit gefächertes Anwendungsspektrum. Dementsprechend hoch sind die Anforderungen, die an die Steuerung von NPM-Maschinen gestellt werden. Bildet sie doch den zentralen Punkt für Programmierung, Abarbeitung und Auswertung von Messvorgängen.

Jedes der Anwendungsszenarien für NPM-Maschinen verlangt von der Steuerung spezielle Funktionen. So unterscheidet sich beispielsweise die Programmierung einer Oberflächenmessung grundlegend von der Arbeit mit der Maschine im Diagnosefall. Ebenfalls unterscheiden sich die einzelnen Zyklen eines Messablaufes stark voneinander. Sobald das Messprogramm erstellt und gestartet wurde, möchte der Nutzer beispielsweise immer den aktuellen Bearbeitungsstatus sehen und im Fehlerfall in den Messprozess eingreifen können.

Die Steuerung von NPM-Maschinen muss sich möglichst flexibel auf alle Situationen des jeweiligen Einsatzgebietes anpassen lassen. Deshalb wird ein modulares Steuerungskonzept vorgeschlagen, das je nach Situation entsprechende Module anbietet und zukünftig die Erweiterbarkeit der Steuerung ermöglicht, indem nach Belieben weitere Module integriert werden können. Im Rahmen dieser Dissertation sind bereits einige Module für die Steuerung von NPM-Maschinen entstanden, von denen hier die Komponenten Messprogrammeditor, die realisierten Funktionen eines Befehlsinterpreters und ein Diagnosemodul vorgestellt werden sollen.

Die Basis aller Module bildet ein durch zusätzliche Fenster erweiterbares zentrales Dialog-Fenster. Der Datenaustausch zwischen den Modulen wird durch das Datenformat XML realisiert. Die Vor- oder Nachteile dieser Datenbasis sind im Kapitel 5 nachzulesen.

Die Programmierung aller hier vorgestellten Module erfolgte in der Programmiersprache C# und dem Werkzeug Visual Studio 2003 bzw. Visual Studio 2005.

6.2 Messprogrammeditor

Eines der zentralen Module bei der Arbeit mit der NPM-Maschine ist der Messprogrammeditor. Er ermöglicht den Zugriff auf alle verfügbaren Maschinenbefehle und soll den Nutzer in die Lage versetzen, schnell und effizient Messprogramme zu erstellen, bereits bestehende Programme zu modifizieren oder bewährte Messprogramme zu Neuen zusammenzusetzen. Dabei greift der Editor auf den in Kapitel 5 genauer beschriebenen Maschinenbefehlssatz zurück. Ziel des Messprogrammeditors ist es, dem Nutzer den gesamten Befehlssatz bereitzustellen, damit sich dieser daraus ein Messprogramm zusammenstellen kann. Die Bereitstellung der Befehle erfolgt grafisch durch die Anordnung in Befehlsgruppen in einem Befehlsbaum. Durch klassische, von modernen Betriebssystemen wie Windows XP, bekannte Mausaktionen wie Doppelklick oder „Drag & Drop“ erfolgt die Erstellung und Editierung eines Messprogramms. Das Ergebnis des Messprogrammeditors ist ein Messprogramm in Form einer XML-Datei.

6.2.1 Aufgabe des Editors

Die Aufgabe des Messprogrammeditors besteht in der Erstellung von Mess-, Konfigurations- und Testprogrammen für NPM-Maschinen. Je nach Einsatzfall soll der Nutzer schnell in der Lage sein, entsprechende Programme zu erstellen oder bereits bestehende zu editieren. Aufgrund des umfangreichen Befehlssatzes der Maschine, es sind derzeit 58 Befehle mit einer Vielzahl unterschiedlichster Parameter, muss dem Nutzer ein effektives Werkzeug zum Erstellen solcher Programme angeboten werden.

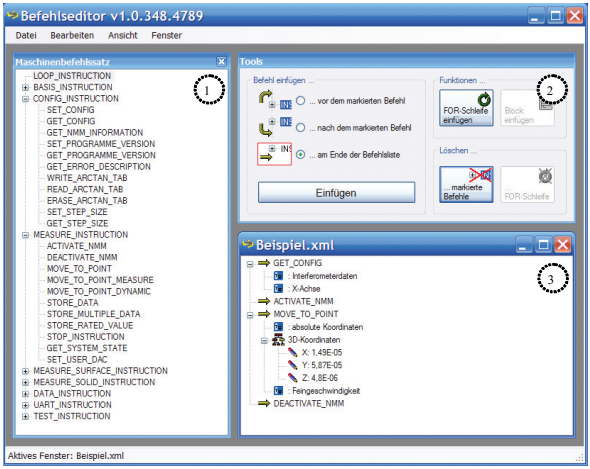


Bild 19: Bedienoberfläche des Messprogrammeditors

Der hier vorgestellte Programmeditor stellt den Befehlssatz in grafischer Weise dar. Die bereitgestellten Befehle repräsentieren den gesamten Befehlssatz, den die NPM-Maschine zur Verfügung stellt. Es ist jedoch auch möglich, je nach Benutzergruppe des Bedieners den Befehlssatz einzuschränken, um die Ausführung bestimmter, kritischer Befehle nur berechtigten Nutzern zu ermöglichen.

Der Befehlssatz wird in einer Baumstruktur, wie im Bild 19 (Fenster 1) zu sehen, dargestellt. Die Anordnung der einzelnen Befehle erfolgt in Befehlsgruppen. Diese leiten sich aus dem entwickelten XML-Befehlssatz ab (siehe Kapitel 5.3). Die einzelnen Befehle können durch Zu- bzw. Aufklappen der einzelnen Baumebenen angezeigt oder ausgeblendet werden. Zu jedem Befehl werden bei Bedarf kurze Hinweistexte eingeblendet. Um eine gute Übersichtlichkeit zu erreichen, werden die Befehlsparameter in dieser Ansicht ausgeblendet.

Zur Auswahl eines Befehls für das zu erstellende Programm stehen mehrere Möglichkeiten zur Verfügung. Zum einen lässt sich ein Befehl durch Doppelklick im Befehlssatz auswählen. Die Anordnung des gewählten Befehls innerhalb des Messprogramms wird dann durch die Auswahl im Fenster 2 vorgenommen. Mögliche Einstellungen sind:

- „... vor dem markierten Befehl“
- „... nach dem markierten Befehl“
- „... am Ende der Befehlsliste“.

Die Wirkung der Auswahl ist intuitiv zu erkennen. In den beiden ersten Fällen muss jeweils ein Befehl ausgewählt, d. h. markiert sein. Ist dies nicht der Fall, wird der Befehl am Ende der Befehlsliste angehängt. Diese Möglichkeit kann auch durch Auswahl der entsprechenden Option, trotz markierter Befehle, eingestellt werden. Die Betätigung des Buttons *Einfügen* führt analog dem Doppelklick zum gleichen Ergebnis. Selbstverständlich ist dazu eine vorherige Auswahl eines Befehls im Maschinenbefehlssatz notwendig. Eine weitere Möglichkeit einen, oder auch mehrere Befehle in ein Messprogramm einzufügen bietet die *Drag & Drop-Funktion*. Damit ist es möglich aus anderen Programmen Befehle „herauszuziehen“ und über dem aktiven Programm „fallen“ zu lassen. Erfolgt das „Fallenlassen“ dabei über einem freien Bereich des Fensters, wird der Befehl am Ende der Liste angehängt. Wird jedoch über einem bestehenden Befehl der Liste „losgelassen“, wird der Befehl entsprechend der Einstellung im Fester 2 (Bild 19) eingeordnet. Schließlich bietet der Button *Block einfügen* die Möglichkeit, ganze XML-Programme an der entsprechenden Stelle in ein Messprogramm einzufügen.

Sich wiederholende Programmbefehle können durch Schleifen mehrfach abgearbeitet werden. Dazu kann mithilfe des Buttons *FOR-Schleife einfügen* eine Schleife programmiert werden. Mehr dazu aber im Abschnitt 6.2.4.

Das Einfügen eines neuen Befehls erfordert natürlich die Angabe erforderlicher Parameter. Dies erfolgt direkt nach der Auswahl des Befehls anhand dynamisch generierter Dialoge, die die Eingabe bzw. Auswahl der Parameter verlangen. Die Parametereingabe wird im Abschnitt 6.2.3 näher erläutert.

Die Entwicklung des Programms ist unmittelbar im Editorfenster zu verfolgen. Änderungen an den Parametern und Befehlen erfolgen in der bereits beschriebenen Art und Weise. Das Löschen einzelner oder mehrerer markierter Befehle ist ebenfalls mittels *Entfernen*-Taste oder dem entsprechenden Button im Fenster 2 (Bild 19) möglich.

Jede Veränderung am Messprogramm wird vom Messprogrammeditor registriert und führt nach Beendigung der Bearbeitung zur Aufforderung zum Speichern. Eine Autospeichern-Funktion erhöht die Sicherheit vor Datenverlust. Alle erzeugten Programme werden im XML-Format gespeichert. Zum genaueren Aufbau einer fertigen Messprogrammdatei sind im Kapitel 6.2.6 nähere Informationen zu finden.

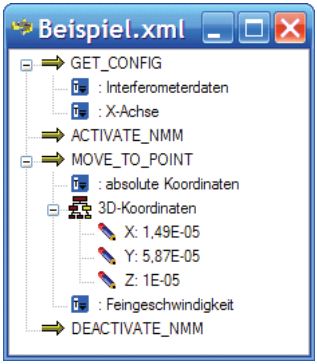


Bild 20: Beispiel für ein einfaches Messprogramm

Bild 20 zeigt eine vergrößerte Darstellung des Fensters 3 im Bild 19. Hier entfällt die Zuordnung zu Befehlsgruppen. Denn hier liegt der eigentliche Informationsgehalt in der Übersicht über die Abfolge der einzelnen Befehle, nicht in deren Anordnung innerhalb des Befehlssatzes. Ebenfalls von zentraler Bedeutung in dieser Ansicht, ist die Darstellung der einzelnen Parameter und ihrer aktuellen Werte. Durch Zu- bzw. Aufklappen der einzelnen Parameterebenen ist so ein schneller Überblick über eingestellte Parameter möglich.

Weiterhin können die Parameter jederzeit direkt in der Befehlsliste editiert werden. Dazu genügt ein Doppelklick, entweder auf einen Befehl, dann werden alle Parameter editiert, oder auf einen einzelnen Parameter, hierbei wird lediglich der gewählte Parameter verändert.

6.2.2 Zugriff auf den Maschinenbefehlssatz

Der ursprüngliche Maschinenbefehlssatz wurde bereits in Kapitel 2.2.2 beschrieben. Die Umsetzung in das Dateiformat XML wurde in Kapitel 5.3 erläutert. Der Messprogrammeditor basiert ausschließlich auf dem in XML vorliegenden Maschinenbefehlssatz. Dazu wird dieser vollständig als Objekt in den Arbeitsspeicher geladen. Dies erfolgt durch ein Objekt vom Typ *XmlDocument*:

```
XmlDocument instructionSetDocument = new XmlDocument();
instructionSetDocument.Load(instructionSetFileName);
```

Dadurch wird das XML-Dokument im Speicher als Baumstruktur abgespeichert. Die Schnittstelle, um auf die einzelnen Zweige des Dokumentes zuzugreifen, heißt *Document Object Model* (DOM) und wird vom *World Wide Web Consortion* (W3C) definiert. Die Klasse *XmlDocument* liefert eine Menge an Funktionen und Eigenschaften, mit deren Hilfe das DOM komfortabel bearbeitet und ausgewertet werden kann. Allerdings ist dieser Vorteil mit einem erheblichen Plus an Speicherplatzbedarf verbunden. In [15] wird etwa von Faktor 100 gesprochen. Mit der derzeitigen Dateigröße von 50KByte der Befehlssatzdatei und der daraus resultierenden Größe im Speicher vom 5MByte ist bei modernen Rechnersystemen allerdings nicht mit Speicherproblemen zu rechnen.

Das geladene Dokument wird zunächst als Liste der zur Verfügung stehenden Befehle abgebildet. Dazu durchlaufen zwei ineinandergeschachtelte foreach-Schleifen den entsprechenden „INSTRUCTIONS“-Zweig der Datei. Die Verzweigung der Baumstruktur der Befehlssatzliste ist mit der der Verschachtelung innerhalb des XML-Dokumentes identisch. So ist eine Zuordnung eines ausgewählten Befehles zu dem entsprechenden Element aus dem DOM möglich.

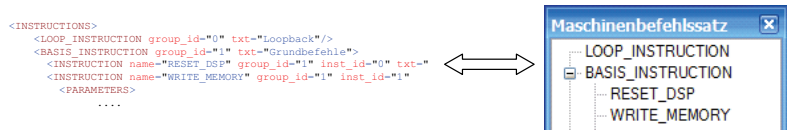


Bild 21: Zuordnung von Befehlssatzdatei und Befehlsliste

Im Unterschied dazu sind in der Befehlsliste des Messprogramms alle Befehle mit der zugehörigen Befehls- und Gruppennummer hinterlegt (siehe Bild 21 und Kapitel 5.3). Auch

hierdurch ist eine eindeutige Zuordnung von Befehlssatzdatei und ausgewähltem Befehl gewährleistet. Mit dem Wissen über Befehlsgruppe und –nummer kann der entsprechende Befehlsknoten („INSTRUCTION“) aus der Befehlssatzdatei ausgewertet werden. Er beinhaltet alle Informationen über Parameter und Rückgabewerte. Diese Informationen werden dazu genutzt entsprechende Dialoge zu generieren, die den Nutzer dazu auffordern, die Werte der Parameter einzugeben bzw. auszuwählen. Die dynamische Generierung von Dialogen anhand der Parameterliste eines Befehls wird im folgenden Abschnitt erläutert.

6.2.3 Generierung dynamischer Eingabedialoge

Die einzelnen Befehle des Maschinenbefehlssatzes besitzen sehr unterschiedliche Strukturen. Einerseits gibt es Befehle, die keine Parameter erwarten. Hier genügt es, lediglich Befehlsgruppe und -nummer an die Maschine zu übergeben, um die Maschine anzusteuern. Zum anderen benötigt die Mehrzahl der Befehle verschiedene Parameter, um entsprechende Funktionsaufrufe der NPM-Maschine zu realisieren. Art und Anzahl der einzelnen Parameter sind dabei von Befehl zu Befehl unterschiedlich. So existieren beispielsweise Parameter, die einfache Wertetypen repräsentieren oder auch solche, die komplexe Strukturen abbilden.

Um die Eingabe der Parameter während der Messprogrammerstellung so komfortabel wie möglich zu machen, wird anhand der Klasse des Parameters eine Eingabemaske in Form eines Windows-Formulars erzeugt. Das Aussehen des Dialoges wird dabei dynamisch dem jeweiligen Parameter angepasst. Dafür sind mehrere Schritte notwendig, um zu derartigen Dialogen zu gelangen.

Zunächst wird der gesamte Befehl mithilfe einer Select-Abfrage als ein XML-Knoten aus dem DOM der Befehlssatzliste ermittelt. Dazu wird ein Suchstring in der Abfragesprache *XPath* [9] benötigt. Folgende Abfrage würde den Befehl *RESET_DSP* liefern:

```
SearchString = "//INSTRUCTIONS/*[position()=2]/*[position()=1]";
XmlNode Inst = InstructionSetDocument.SelectSingleNode(SearchString);
```

Die einzelnen Positionen innerhalb der Befehlssatzdatei werden anhand des selektierten Befehls in der Befehlsliste des Messprogrammeditors gefunden.

Ist der Befehlsknoten ermittelt, wird geprüft, ob der Befehl Parameter erwartet. Dies wird über das Vorhandensein von Knoten namens *PARAM* festgestellt:

```
XmlNodeList ParaList = Inst.SelectNodes("./PARAMETERS/PARAM");
if (ParaListe.Count > 0) ...
```

Sind keine Parameter erforderlich kann der selektierte Befehl ohne weitere Eingaben in die Befehlsliste des Messprogramms übernommen werden. Andernfalls ist die Generierung von Dialogen zur Parametereingabe erforderlich. Dies erfolgt für jeden in der Liste *ParaList* gespeicherten Knoten.

Für jeden Parameter ist zunächst der Typ zu prüfen. Dabei werden drei Arten unterschieden:

1. Auswahltypen (Enumerationen):
   Derartige Parameter bieten dem Nutzer eine Auswahl vordefinierter Werte. Auswahlparameter haben jeweils einen eigenen Typ und ein entsprechendes Element in der XML-Befehlssatzdatei. Anhand des Parametertyps und dem Vorhandensein eines solchen Elementes innerhalb des Blockes *SYMBOLS*, können die möglichen Auswahlwerte für einen derartigen Parameter ermittelt werden.

2. Strukturtypen:
   Strukturparameter bestehen aus einer Liste mehrerer Werte von (meist einfachen) Wertetypen. Die einzelnen Komponenten einer Parameterstruktur sind ebenfalls in der Befehlssatzdatei hinterlegt. Hierfür ist der Block *STRUCTURES* zuständig.

3. Wertetypen:
   Wertetypen sind Einzelwerte für Parameter. Sie bestehen in der Regel aus Ganzzahl- oder Gleitkommawerten. In einigen Fällen sind auch Felder eines Datentyps zu übergeben.

Anhand der Eingruppierung des Parameters in eine der besprochenen Arten unterscheidet sich die Generierung der jeweiligen Dialoge etwas voneinander. Gleich ist bei allen Varianten aber, dass die Dialoge auf der Verwendung eines *DataGridView*-Steuerelements beruhen. Dies hat den Vorteil, dass in einzelnen Spalten die Bezeichnung des Wertes, der Wert an sich und eine kurze Beschreibung angezeigt werden können. Je nach Anzahl der Parameter werden entsprechend viele Zeilen erzeugt.

Für Auswahlparameter ergeben sich lediglich zwei Spalten, wie in Bild 22 zu sehen ist. Diese repräsentieren die zur Verfügung stehenden Ausprägungen des Parameters und eine kurze Beschreibung desselben. Die Auswahl eines Wertes erfolgt durch Doppelklick auf die jeweilige Zeile. Damit wird die Auswahl übernommen und der Parameter in die Befehlsliste des Messprogramms eingetragen.

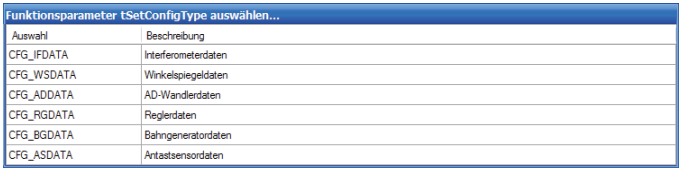


Bild 22: Auswahldialog eines Auswahlparameters

Ein etwas anderes Aussehen weisen die Eingabedialoge für Struktur- und Wertetypparameter auf. Hier werden neben dem Namen und einer kurzen Beschreibung zusätzlich der Datentyp und der Wert des Parameters als Spalte aufgeführt. Die Übernahme der Daten erfolgt durch Betätigung der Schaltfläche *OK*.

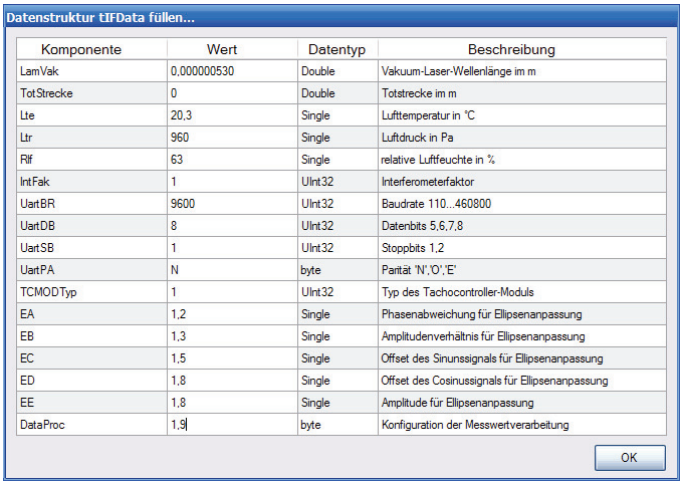


Bild 23: Eingabedialog eines Strukturparameters

Bei Verwendung eines DataGridView-Steuerelements kann für jede Zelle der Tabelle der Datentyp des in ihr einzugebenden Wertes festgelegt werden. Hierdurch wird erreicht, dass schon während der Eingabe eine Überprüfung der Werte vorgenommen wird. Auf entsprechende Ausnahmen kann dementsprechend reagiert und dem Nutzer ein Hinweis über die Fehleingabe angeboten werden.

Da einige Auswahlparameter wieder neue Parameter nach sich ziehen wird geprüft, ob der gerade editierte Parameter selbst Parameter erfordert. Ist dies der Fall, wird die Generierung der entsprechenden Eingabedialoge rekursiv wiederholt.

6.2.4 Befehle zur Ablaufsteuerung

Einige Befehle eines Messprogramms wiederholen sich häufig. So wird beispielsweise bei einem Flächenscan einer Oberfläche zunächst das Höhenprofil einer Linie gemessen. Anschließend wird das Messobjekt um eine definierte Strecke verschoben und eine neue Messung neben der gerade gemessenen Linie durchgeführt. Hierbei können die Bewegung des Messtisches während der Messung und die beim Verschieben des Messobjektes als Relativbewegung zum jeweiligen Ausgangspunkt betrachtet werden. Daher ist es für die Erstellung des Messprogramms eine Erleichterung, wenn diese Relativbewegung in einer Schleife programmiert werden kann.

Die Programmierung einer derartigen Schleifenstruktur ermöglicht der Messprogrammeditor durch das Einfügen einer *FOR-Schleife*. Einziger Eingabewert des Nutzers ist die Anzahl der Wiederholungen. Innerhalb einer FOR-Schleife lassen sich beliebige Befehle anordnen. Die Anzahl der Wiederholungen ist jederzeit im Editor editierbar. Ein Löschen der Schleife ist ebenfalls möglich. Hierbei kann der Nutzer wählen, ob lediglich die Schleife selbst oder aber der gesamte von der Schleife umschlossene Befehlszweig gelöscht werden soll.

Der Interpreter muss die Schleifenstruktur erkennen und bei der Abarbeitung des Messprogramms entsprechend oft die gewünschten Befehle mit den entsprechenden Parametern ausführen. Dazu wird im Messprogramm eine FOR-Schleife durch einen Knoten *FOR* repräsentiert.

Weitere denkbare Einsatzfälle für Ablaufsteuerungen sind Elemente wie *IF-ELSE* oder auch *WHILE*. In der jetzigen Realisierung des Messprogrammeditors sind diese aber nicht aufgenommen. Für einen weiteren Ausbau des Editors sind sie aber vorgesehen.

6.2.5 Plausibilitätsprüfung

Der Messprogrammeditor bietet autorisierten Nutzern den gesamten Befehlssatz an. Das Auswahlkonzept der einzelnen Befehle kann aber unter Umständen dazu führen, dass Befehle in einer Reihenfolge gewählt werden, die eine Abarbeitung des Messprogramms unmöglich machen. Weiterhin ist es denkbar, dass Parameter eingegeben wurden, deren Werte nicht umsetzbar sind. Denkbar wäre hier ein Verfahrbefehl an eine Position, die außerhalb des Bewegungsbereiches der Maschine liegt. Solche Fehleingaben sollten vom Messprogrammeditor erkannt und durch geeignete Hinweise und Hilfsangebote an den Nutzer behoben oder korrigiert werden können.

Die aktuelle Implementierung des Messprogrammeditors realisiert eine erste Variante der Plausibilitätskontrolle. Sie beinhaltet eine Überprüfung des fertigen Messprogramms auf eine korrekte Reihenfolge der *Aktivierung* der Maschine und Befehlen, die eine Aktivierung erfordern. Diese Überprüfung erfolgt durch eine Gültigkeitsprüfung der XML-Messprogrammdatei gegen das dazugehörige Schema. Eine weitere Verbesserung der Plausibilitätsüberprüfung ist in einer weiteren Entwicklungsphase des Editors vorgesehen. Dazu ist eine Erweiterung des Maschinenbefehlssatzes notwendig, der beispielsweise eine Implementierung von Maximalwerten für Parameter beinhaltet.

6.2.6 Aufbau und Schema der Messprogrammdatei

Das Ergebnis einer Programmierung mit dem Messprogrammeditor ist eine XML-Datei. Sie beinhaltet alle Informationen die notwendig sind, um die NPM-Maschine zu steuern. Im folgenden Beispiel ist die Befehlsliste eines einfachen Messprogramms dargestellt. Das Messprogramm dient lediglich dazu den Aufbau der Messprogrammdatei zu verdeutlichen. Das Programm beinhaltet den Messbefehl *MOVE_TO_POINT_MEASURE*. Dieser veranlasst die Maschine eine Messung entlang eines angegebenen Vektors durchzuführen. Der Vektor wird durch die Struktur *tDouble3d* angegeben, der Abstand zwischen den Messpunkten gibt der dritte Parameter an. Durch die umgebende FOR-Schleife und der relativen Koordinatenangabe (*tMoveMode* = 1 = relative Koordinaten) wird die Messung zehnmal wiederholt. Dabei werden zehn Messungen entlang der X-Achse durchgeführt.

```
<?xml version="1.0" encoding="utf-8"?>
<PROCESSING_SEQUENCE>
  <FOR iterations="10">
    <INSTRUCTION  name="MOVE_TO_POINT_MEASURE"  group_id="17"  inst_id="3"
          txt="Bewegung zu Punkt auf Linie mit Messwerterfassung (Nutzer)">
      <PARAM>
        <SYMBOL type="tMoveMode" value="1" />
        <STRUCT type="tDouble3d">
          <COMPONENT name="X" start_byte="4" type="Double" value="0.001" />
          <COMPONENT name="Y" start_byte="12" type="Double" value="0.0" />
          <COMPONENT name="Z" start_byte="20" type="Double" value="0.0" />
        </STRUCT>
        <SYMBOL type="Double" value="0.000001" />
      </PARAM>
      <ERR_VAL type="byte" start_byte="2" />
      <RET_VAL type="int" start_byte="3" />
    </INSTRUCTION>
  </FOR>
</PROCESSING_SEQUENCE>
```

Einige Informationen sind für die Abarbeitung des Messprogramms nicht zwingend erforderlich. Beispielsweise könnten die Attribute *name* und *txt* des *INSTRUCTION*-Knotens weggelassen werden. Der Befehl kann anhand der Gruppen- und Befehlsnummer eindeutig identifiziert werden. Die Informationen können aber dem Bediener das Lesen der Datei erleichtern.

Die Möglichkeit des Lesens von Messprogrammdateien in anderen Programmen, z. B. einem Texteditor, birgt aber auch die Gefahr, den Nutzer dazu zu verleiten, Änderungen an der XML-Datei vorzunehmen. Dabei ist es durchaus denkbar, dass fehlerhafte Messprogrammdateien entstehen. Dabei können sowohl Fehler in der Wohlgeformtheit des Dokumentes, als auch in der Syntax der Messbefehle auftreten. Beide Fehler haben unterschiedliche Auswirkungen auf die Ausführbarkeit des Messprogramms. Fehler in der Wohlgeformtheit des Dokumentes sind Fehler im Aufbau der XML-Datei. Eine derart fehlerhafte Datei ist im eigentlichen Sinne keine XML-Datei. Fehler dieser Art sind aber durch entsprechende Klassen des .NET Framework (siehe Kapitel 7.1.3.1) leicht erkennbar. Dem Nutzer können somit entsprechende Meldungen angezeigt werden. Schwieriger ist das Erkennen von Fehlern in der Syntax einzelner Messbefehle. Fehlen beispielsweise Parameter, so ist das Dokument zwar eine wohlgeformte Datei, im Sinne eines Messprogramms aber fehlerhaft. Für diese Art von Fehlern gibt es innerhalb der XML-Familie eine Möglichkeit diese zu erkennen. Dazu werden XML-Schemata verwendet. Ein XML-Schema gibt die Struktur einer Datei vor. So kann festgelegt werden, welche Elemente die Datei beinhalten muss, welche Reihenfolge die einzelnen Elemente haben müssen und wie oft beispielsweise ein Element vorkommen darf. Mithilfe von XML-Schemata kann also genau festgelegt werden, wie das Messprogramm aufgebaut sein muss. Ein Schema für Messprogrammdateien ist in Bild 24 zu sehen. Mithilfe dieses Schemas können erstellte Messprogramme auf ihre Gültigkeit hin überprüft werden. Damit kann sicher gestellt werden, dass keine fehlerhaften, außerhalb des Messprogrammeditors editierten, Messprogramme zur Abarbeitung gelangen.

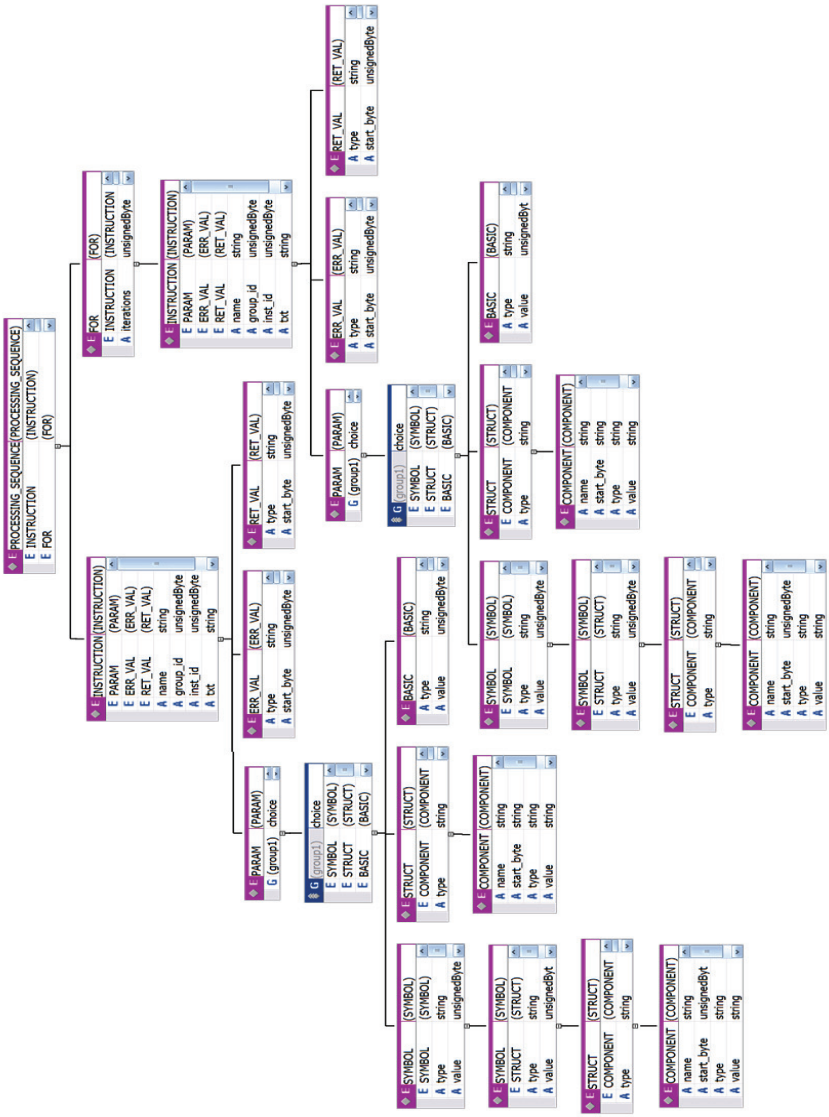


Bild 24: XML-Schema für Messprogrammdateien

6.3 Diagnosemodul

Um jederzeit die korrekte Funktionsfähigkeit einer NPM-Maschine feststellen zu können, ist ein Diagnosemodul erforderlich. Somit können mithilfe von Informationen über Antriebe, Interferometer oder eingestellte Reglerparameter Rückschlüsse auf Fehlfunktionen der Maschine gezogen werden. Eine modulare Steuerung von NPM-Maschinen muss deshalb ein Diagnosemodul vorsehen. Die Aufgaben eines solchen Diagnosemoduls umfassen dabei die Ermittlung der gesuchten Maschinenparameter, die Anzeige dieser Werte in geeigneter Art und Weise, die Bereitstellung von Funktionen die Maschinenaktionen bewirken (bspw. das Verfahren des Messtisches um Fehler des Antriebssystems untersuchen zu können) und schließlich die Möglichkeit Parameterwerte gezielt zu verändern.

Dem Diagnosemodul als eine Komponente der Steuerung von NPM-Maschinen kommt dabei in besonderem Maße eine Sonderstellung zu. Die Benutzung dieser Komponente wird vorrangig nicht vom Bedien-PC der Maschine her, sondern aufgrund der besonderen Aufgabenstellung vorzugsweise auch von entfernten Orten erfolgen. Neben den eigentlichen Funktionen des Zugriffes auf die Maschine muss das Diagnosemodul daher auch Teleservice-Funktionalität bereitstellen.

Im Rahmen dieser Dissertation und einer Diplomarbeit ist ein eigenständiges Diagnosemodul entstanden, welches in das Gesamtkonzept einer Maschinensteuerung integriert werden kann. Die realisierten Möglichkeiten und den Aufbau des Diagnosemoduls beschreiben die folgenden Abschnitte.

6.3.1 Darstellbare Informationen

Das realisierte Diagnosemodul verwendet zur Erfassung von Maschineninformationen den Befehl *GET_CONFIG*, der im Befehlssatz der Maschine zur Verfügung steht. Mithilfe des Parameters *tGetConfigType* liefert der Befehl Informationen über folgende Komponenten der NPM-Maschine:

- Interferometerdaten: Informationen über den Zustand der Interferometer zur Positionsbestimmung des Messtisches
- Winkelspiegeldaten: Informationen über Korrekturfunktionen der Winkelspiegel
- A/D-Wandlerdaten: Informationen über Korrekturkoeffizienten der A/D-Wandler zur Messwerterfassung

- Reglerdaten: Informationen über die Regler der Antriebe
- Bahngeneratordaten: Informationen über die Parameter der Bahngeneratoren
- Antastsensordaten: Informationen über die Antastsensoren

Die Auswahl der gesuchten Information erfolgt durch den Parameter tGetConfigType. Je nach Wert des Parameters wird die entsprechende Bytefolge an die Maschine gesendet, dort ausgewertet und die Antwort als Bytefeld zurückgeben. Das Bytefeld muss dann wiederum ausgewertet und dargestellt werden.

Für das Verändern der Werte gibt es ebenfalls eine bereits implementierte Maschinenfunktion. Sie heißt *SET_CONFIG* und ist als Gegenstück zu GET_CONFIG zu sehen. Alle Aussagen, die im Folgenden auf GET_CONFIG angewendet werden, sind in umgekehrter Art und Weise auch für SET_CONFIG gültig.

Interferometerdaten

Die Interferometer dienen wie bereits beschrieben der Erfassung der Position des Messtisches durch Messung des Abstandes zwischen Interferometer und der am Messtisch angebrachten Spiegelecke. Bei der Verwendung des Befehls GET_CONFIG zur Bestimmung der Interferometer-Parameter tritt eine Besonderheit auf, da der Parameter tGetConfigType selbst einen weiteren Parameter benötigt. Dieser zusätzliche Parameter gibt an, für welche Achse die Interferometerdaten bestimmt werden sollen. Die Antwort auf diese Anforderung ist ein 70 Byte großes Feld mit Informationen über Wellenlänge, Temperatur oder Luftdruck (siehe Bild 25). Um Informationen über die Interferometer aller sechs Achsen zu erhalten, ist GET_CONFIG sechsmal aufzurufen.

Winkelspiegeldaten

Die Spiegelecke ist ein wichtiges Bauelement des Messsystems der NPM-Maschine. Über den gesamten Arbeitsraum einer Achse reflektiert eine Ebene der Spiegelecke das Licht des entsprechenden Interferometers. Die durch die Fertigung der Spiegelecke unumgänglichen Abweichungen von einer idealen ebenen Oberfläche werden durch Korrekturfunktionen ausgeglichen. Die Koeffizienten dieser Funktionen werden für die Achsen X, Y und Z direkt im Speicher des DSP abgelegt. Durch den Aufruf dieser Informationen können die eingestellten Koeffizienten überprüft werden.

A/D-Wandlerdaten

Der Bereich der Wandlerdaten liefert die Koeffizienten der Kennlinienfunktion der A/D-Wandler. Mithilfe dieser Informationen können Rückschlüsse auf die Ermittlung der Werte für die Achsen X, Y, und Z sowie die Verkippung um die einzelnen Achsen gezogen werden.

Reglerdaten

Die in der DSP-Einheit der Maschine hinterlegten Reglerdaten bestimmen die Eigenschaften der Antriebsregelung in den Achsen X, Y und Z sowie die Regelung der Antriebe für die Kompensation der Verkippung. Dadurch lassen sich hiermit die Reglerparameter beurteilen bzw. gegebenenfalls beeinflussen.

Bahngeneratordaten

In den Bahngeneratordaten sind Informationen über die Kinetik der Maschine eingearbeitet. So ist hier beispielsweise ersichtlich, wie stark die Antriebe in den jeweiligen Geschwindigkeitsstufen beschleunigen, wie groß die Messunsicherheit der Interferometer und Winkelsensoren ist oder wie groß der maximale Positionsfehler ist.

Antastsensordaten

Hier sind Informationen über das eingesetzte Antastsystem abzulesen. Die Rückgabe erlaubt Rückschlüsse auf den Sensortyp und das Antastverhalten.

Bild 25: Parameter und Rückgabewerte der Funktion *GET_CONFIG*

6.3.2 Realisierung als Webdienst

Die beschriebenen Konfigurationsbefehle, GET_CONFIG zum Lesen bzw. SET_CONFIG zum Schreiben der Konfiguration, bilden die Basis für ein Windows-Programm, das als Diagnose- und Konfigurationsmodul dienen soll. Die Bedienoberfläche des in C# programmierten Programms ist in Bild 26 zu sehen. Der umfangreiche Informationsgehalt der Konfigurationsbefehle macht es notwendig, die Informationen auf mehrere Registerkarten zu verteilen. So gibt es entsprechende Registerkarten zur Dar- und Einstellung der einzelnen Parameter für die Interferometer, den Winkelspiegel, den Regler, usw. Im Anhang sind die Darstellungen der anderen Registerkarten der Bedieneroberfläche zu sehen.

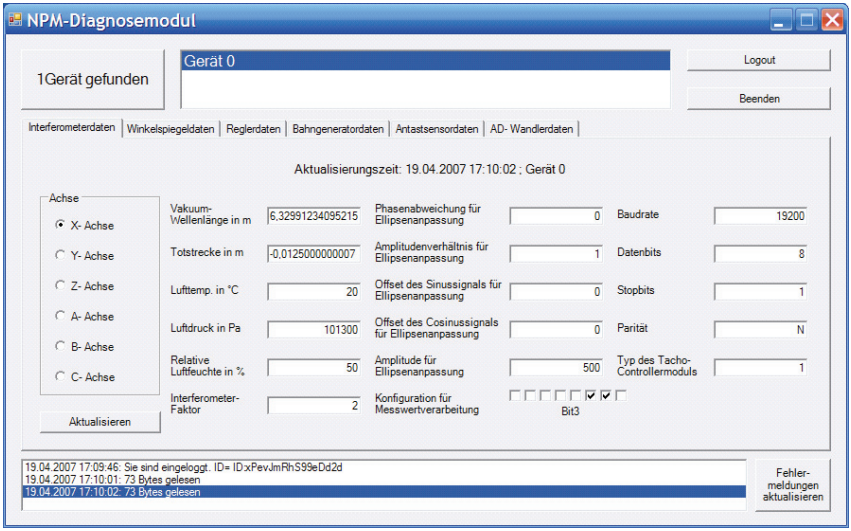


Bild 26: Bedienoberfläche des Diagnosemoduls

Da der eigentliche Nutzen des Diagnosemoduls erst durch Teleservicefunktionalität, also durch Diagnose und Konfiguration einer entfernten Maschine, deutlich wird, wurde dieses Programmmodul als verteilte Anwendung in Form eines Webdienstes erstellt. Dazu wurden einige Methoden der Programmklasse als Webmethoden deklariert und die Klasse der Methoden mit dem Attribut *WebService* versehen. Die genaue Funktionsweise von Webdiensten wird im Kapitel 7 angegeben.

Um die Sicherheit der Maschine zu gewährleisten, sind die Methoden dieses Webdienstes nur nach erfolgreicher Authentifizierung nutzbar. Dazu wurden Untersuchungen durchgeführt, auf welche Art und Weise Webdienste gegen unberechtigten Zugriff geschützt werden können.

Einige Ergebnisse sind in einer Diplomarbeit zusammengefasst worden [19]. Als Fazit der Diplomarbeit lässt sich ableiten, dass das .NET-Framework drei integrierte Methoden zur Authentifizierung anbietet:

- Formularbasierte Authentifizierung
- (Microsoft) Passport-Authentifizierung
- Windows-Authentifizierung

Die Formularbasierte Authentifizierung ist für die Verwendung in ASP.NET-Webseiten vorgesehen. Sie basiert auf html-Formularen zur Eingabe von Nutzername und Passwort, den sogenannten *Credentials* und der Verwendung von *Cookies*. Da der Client des Diagnosemoduls in Form eines Windowsprogramms erstellt werden sollte, konnte diese Variante nicht eingesetzt werden. Auch die Passport-Authentifizierung ist nur für Webseiten vorgesehen. Hier kommt aber dazu, dass der Webdienst zunächst von Microsoft zertifiziert wird. Dieser kostenpflichtige Vorgang ist notwendig, um einen Zugangsschlüssel für den Passport-Dienst von Microsoft zu erhalten. Nur so ist möglich, den Dienst zu nutzen. Diese Variante schied daher ebenfalls aus. Die dritte, integrierte Variante ist die Windows-Authentifizierung. Diese verwendet die in Windows integrierte Benutzerverwaltung. Die Benutzerverwaltung kann so konfiguriert werden, dass die Anmeldedaten des angemeldeten Benutzers an die Internet Information Services (IIS) gesendet werden. Diese werden dann zur Authentifizierung gegenüber dem Webdienst genutzt. Nachteilig an diesem Verfahren ist aber, dass für jeden Nutzer des Webdienstes ein eigenes Konto auf dem Server eingerichtet werden müsste. Aus diesem Grund wurde auch dieses Verfahren nicht eingesetzt.

Da keiner der integrierten Authentifizierungsmechanismen verwendet werden konnte, muss eine eigene Authentifizierungsmethode erstellt werden. Dazu wurde der Übertragungsmechanismus von Webdiensten genutzt. Webdienste senden ihre Daten in Form von XML-Daten. Diese XML-Dokumente besitzen ein spezielles Protokoll, das *Simple Object Access Protocol* (SOAP). In diesem Protokoll ist ein *Header* vorgesehen, der beispielsweise auch die Übertragung von Nutzerinformationen beinhalten kann. Das .NET Framework stellt leistungsfähige Klassen bereit, die diese Header-Informationen auswerten können. Die Zugriffsteuerung und die Authentifizierung mittels SOAP-Header erfordern folgende Schritte:

- Anlegen einer Nutzerdatenbank auf dem Server
  Auf dem Server muss eine Datenbank über die Berechtigungen eines Nutzers entscheiden können. Im Beispiel des Webdienstes für das Diagnosemodul wurde diese

Datenbank als XML-Datei ausgebildet. Diese Datei enthält die Nutzerinformationen, Nutzername und Passwort. Um das Auslesen der Passwörter zu verhindern, wurden diese mittels eines Hash-Algorithmus verschlüsselt abgelegt.

- Authentifizierung
  Der Client muss sich zunächst gegenüber dem Webdienst authentifizieren. Dazu startet das Windows-Programm mit einer Eingabeaufforderung von Nutzername und Passwort. Mit diesen Informationen prüft der Server die Berechtigung des Nutzers. Ist der Nutzer berechtigt den Dienst zu nutzen, erhält er eine zufällig generierte ID, mit der er sich bei allen weiteren Anfragen an den Webdienst ausweisen muss. Diese ID verfällt nach 15min, wenn der Dienst nicht innerhalb dieser Frist genutzt wird.

- Zugriffskontrolle
  Der Server muss sicherstellen, dass immer nur ein Nutzer gleichzeitig die Maschine, bzw. den Webdienst nutzt. Dazu wird beim Einloggen eines Nutzers eine XML-Datei auf dem Server erzeugt, die die Aktionen des eingeloggten Nutzers registriert. Erst wenn der eingeloggte Nutzer ausgeloggt wird oder aufgrund mangelnder Aktivität die Sitzung beendet wird, kann sich ein anderer Nutzer anmelden.

- Abmeldung
  Das erfolgreiche Anmelden eines Nutzers wird in einer XML-Datei vermerkt. Meldet sich der Nutzer ab oder wird wegen einer Zeitüberschreitung die Sitzung abgebrochen, kann diese Datei auf dem Server gelöscht werden. Eine neue Anmeldung wird damit wieder möglich.

In Bild 27 ist der Log-in-Vorgang nochmals dargestellt. Bei erfolgreicher Anmeldung übermittelt der Server zunächst eine ID. Diese ist bei allen weiteren Anfragen an den Webdienst als Parameter zu übergeben. Der Webdienst prüft zunächst die Gültigkeit der ID. Ist die Prüfung erfolgreich, wird die entsprechende Anfrage abgearbeitet und das Ergebnis an den Client zurückgesendet. Dieser kann die Antwort wie gewohnt weiterverarbeiten.

Kritisch ist der Log-in-Vorgang hinsichtlich des Übertragungsweges zu sehen. Hier wird das Passwort einmalig an den Server übertragen. Zwar erfolgt die Übertragung in codierter Form, eine Rekonstruktion des eigentlichen Passwortes ist nicht möglich, aber theoretisch könnte das codierte Passwort ausgespäht und für eine missbräuchliche Anmeldung verwendet werden. Für die zukünftige Nutzung muss noch geprüft werden, ob für den sensiblen Log-in-Vorgang eine verschlüsselte Übertragung zum Einsatz kommen kann oder muss.

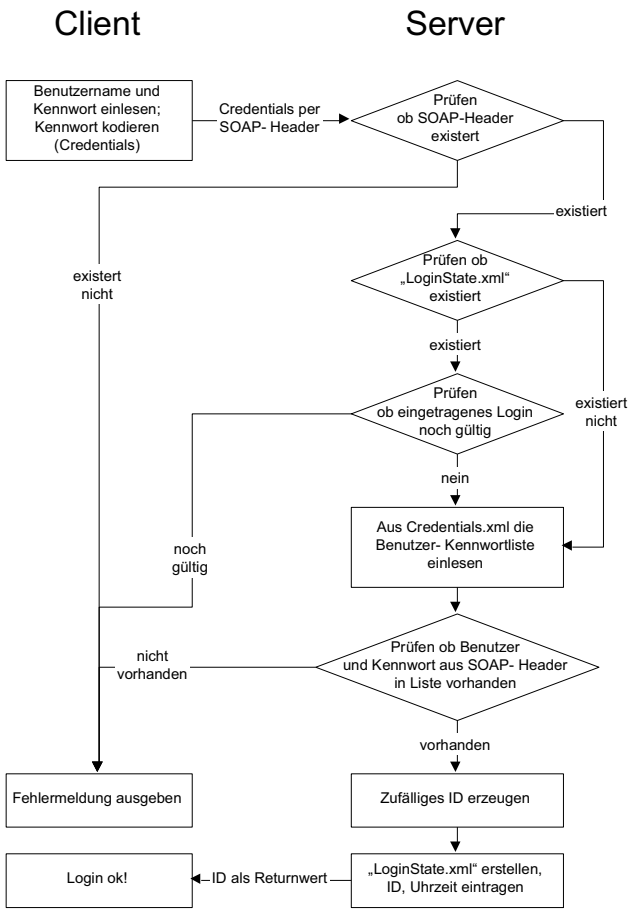


Bild 27: Log-in-Vorgang gegenüber dem Webdienst für das Diagnosemodul [19]

Das Diagnosemodul stellt einen Programmteil dar, der für die Diagnose und die Konfiguration von NPM-Maschinen äußerst wichtig ist. Durch die Aufteilung des Programms in eine Bedieneroberfläche (Client) und einen Webdienst mit direkter Anbindung an die Maschine ist sichergestellt, dass eine Diagnose auch für eine entfernte Maschine realisiert werden kann.

6.4 Messprogramminterpreter

Der Messprogramminterpreter ist das Bindeglied zwischen dem Messprogrammeditor und der Echtzeitsteuerung der NPM-Maschine. Aber auch die Kommunikation zwischen dem Webdienst des Diagnosemoduls und der NPM-Maschine übernimmt der Interpreter. Aufgrund des großen Befehlsumfanges und der großen Zahl unterschiedlichster Parameterkombinationen muss der Interpreter flexibel auf verschiedene Eingaben reagieren können. Weiterhin ist die Realisierung von Funktionen zur Ablaufsteuerung, z. B. die Abarbeitung von Schleifen, Aufgabe des Interpreters.

Im Rahmen dieser Dissertation wurde ein erster Interpreter entwickelt, der die Abarbeitung einer Befehlsfolge realisieren kann. Im Folgenden wird die Funktionsweise dieses Interpreters erläutert.

- Gültigkeitsprüfung mit XML-Schema
  Zunächst wird die Messprogrammdatei auf Gültigkeit geprüft. Dies erfolgt mit dem bereits in Kapitel 6.2.6 angesprochenen XML-Schema. Ist die Datei gültig, kann das Programm abgearbeitet werden.

- Ermittlung des ersten Befehls
  Die Messprogrammdatei wird nach dem ersten XML-Element innerhalb der Befehlsliste *PROCESSING_SEQUENCE* durchsucht. Ist dieses Element eine *FOR*-Schleife, wird eine neue Liste mit den Befehlen innerhalb der Schleifenelemente erstellt. Mit dieser neuen Befehlsliste wird rekursiv die Befehlsabarbeitung erneut gestartet. Wird ein *INSTRUCTION*-Element gefunden, werden die Attribute *group_id* und *inst_id* zur eindeutigen Identifizierung des Befehls ermittelt. Diese beiden Werte werden in das Bytefeld für den Funktionsaufruf geschrieben.

- Bestimmung der Befehlsparameter
  Nach der eindeutigen Bestimmung des Befehls, kann innerhalb der Befehlssatzdatei die Liste der Parameter ermittelt werden. Dazu wird überprüft, ob das Element *PARAMETERS* vorhanden ist. Ist dies der Fall, werden rekursiv alle Parameter bestimmt. Dazu muss zunächst ermittelt werden, um welchen Typ es sich bei dem Parameter handelt (Wertetyp, Auswahltyp, Struktur). Dementsprechend kann das Bytefeld für den Funktionsaufruf mit den Werten aus der Messprogrammdatei gefüllt werden. Bild 29 zeigt das Ablaufdiagramm für die Bestimmung der Parameter in Abhängigkeit der Fälle Funktionsaufruf und Ermittlung der Rückgabewerte.

- Senden des Befehls

  Sind alle Parameter in das Bytefeld eingetragen worden, kann der Befehl abgearbeitet werden. Dazu wird das generierte Bytefeld mithilfe der durch den USB-Treiber bereitgestellten Funktionen an die Maschine gesendet.

- Lesen der Maschinenantwort

  Die Maschine quittiert die erfolgreiche Interpretation des Befehles ebenfalls mit einem Bytefeld. Dieses muss über den entsprechenden Lesebefehl des Treibers ausgelesen werden. Byte 2 gibt Aufschluss über den Erfolg des Schreibbefehls. Nur wenn dieses Byte den Wert 0 hat, ist der Befehl korrekt von Maschine interpretiert worden. In diesem Fall kann die Abarbeitung fortgesetzt werden. Im Fehlerfall muss eine entsprechende Reaktion, zum Beispiel eine Fehlermeldung, ausgelöst werden.

- Interpretation der Rückgabewerte

  Der Interpreter muss prüfen, ob der ausgeführte Befehl weitere Rückgabewerte besitzt. Dazu dient das Element *RETURN_VALUES* innerhalb der Befehlssatzdatei. Diese werden wiederum rekursiv auf den jeweiligen Typ geprüft und entsprechend aus dem Antwort-Bytefeld ausgelesen.

- Überprüfung des Maschinenzustandes

  In einigen Fällen ist der Befehl mit der Übermittlung und erfolgreichen Interpretation durch die Maschine vollständig abgearbeitet. Dies ist beispielsweise bei dem Befehl GET_CONFIG so. Hier werden die angeforderten Informationen direkt im Antwort-Bytefeld zurückgesendet und können entsprechend weiterverarbeitet werden. In anderen Fällen bedeutet die erfolgreiche Interpretation aber nur das Anstoßen eines Vorganges. Beispielsweise führt die Abarbeitung des Befehls MOVE_TO_POINT dazu, dass die durch die Parameter übergebene Position angefahren wird. Die Antwort der Maschine gibt aber nur über den erfolgreichen Start der Funktion, nicht aber über das Erreichen der Zielposition Auskunft. Der Interpreter muss also prüfen, ob ein bestimmter Maschinenzustand erreicht ist, bevor der nächste Befehl abgearbeitet werden kann. Diese Information kann aus dem Element *READY_IF* ausgelesen werden. Die Verarbeitung dieses Elementes ist derzeit nur rudimentär umgesetzt. Da dieses Element in der Regel weitere Befehle nach sich zieht, z. B. der Aufruf des Befehls GET_SYSTEM_STATE, sind hier weitere Überarbeitungen des XML-Befehlssatzes notwendig.

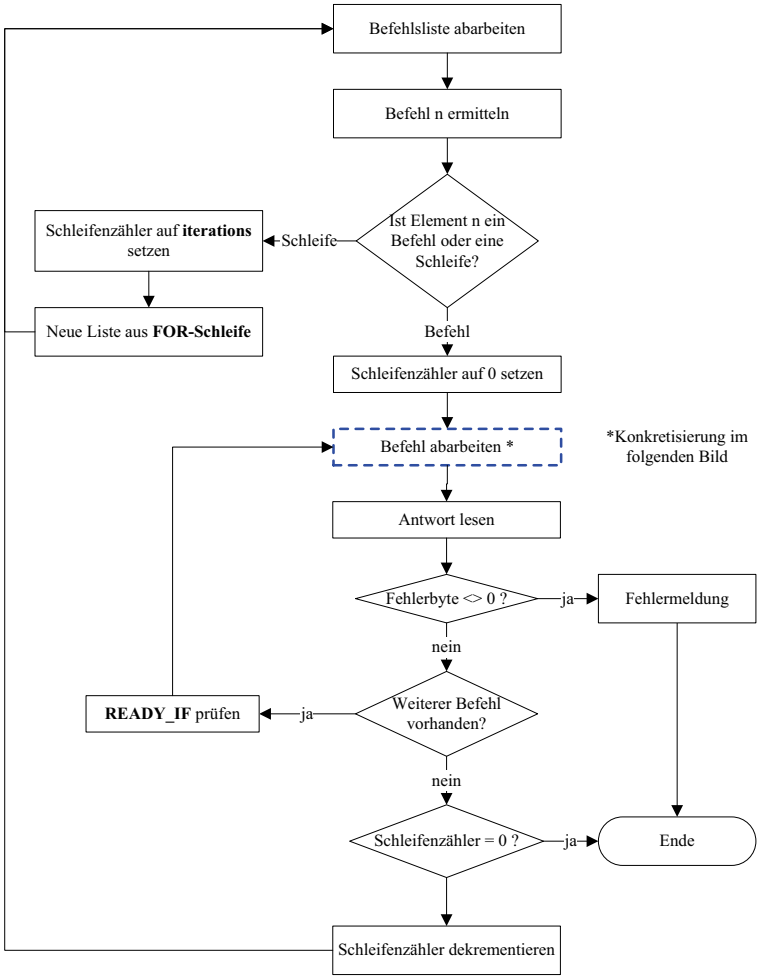


Bild 28: Abarbeitung einer Befehlsliste aus einer XML-Messprogrammdatei für NPM-Maschinen

In Bild 28 ist der Ablauf einer Abarbeitung einer Befehlsliste schematisch dargestellt. Um den Befehl abzuarbeiten, müssen die in der Messprogrammdatei gespeicherten Parameter bestimmt werden. Aufgrund der Vielzahl unterschiedlicher Parameter ist dieser Vorgang sehr komplex. In Bild 29 wird der Vorgang der Parameterbestimmung schematisch dargestellt. Dieser Vorgang ist für das Diagnosemodul erfolgreich implementiert worden.

Der Interpreter bildet wie eben beschrieben einen zentralen Punkt innerhalb der Maschinensteuerung. Aufgrund seiner Komplexität unterliegt er ständigen Änderungen. Über zukünftige Aufgaben gibt das Kapitel 8 Aufschluss.

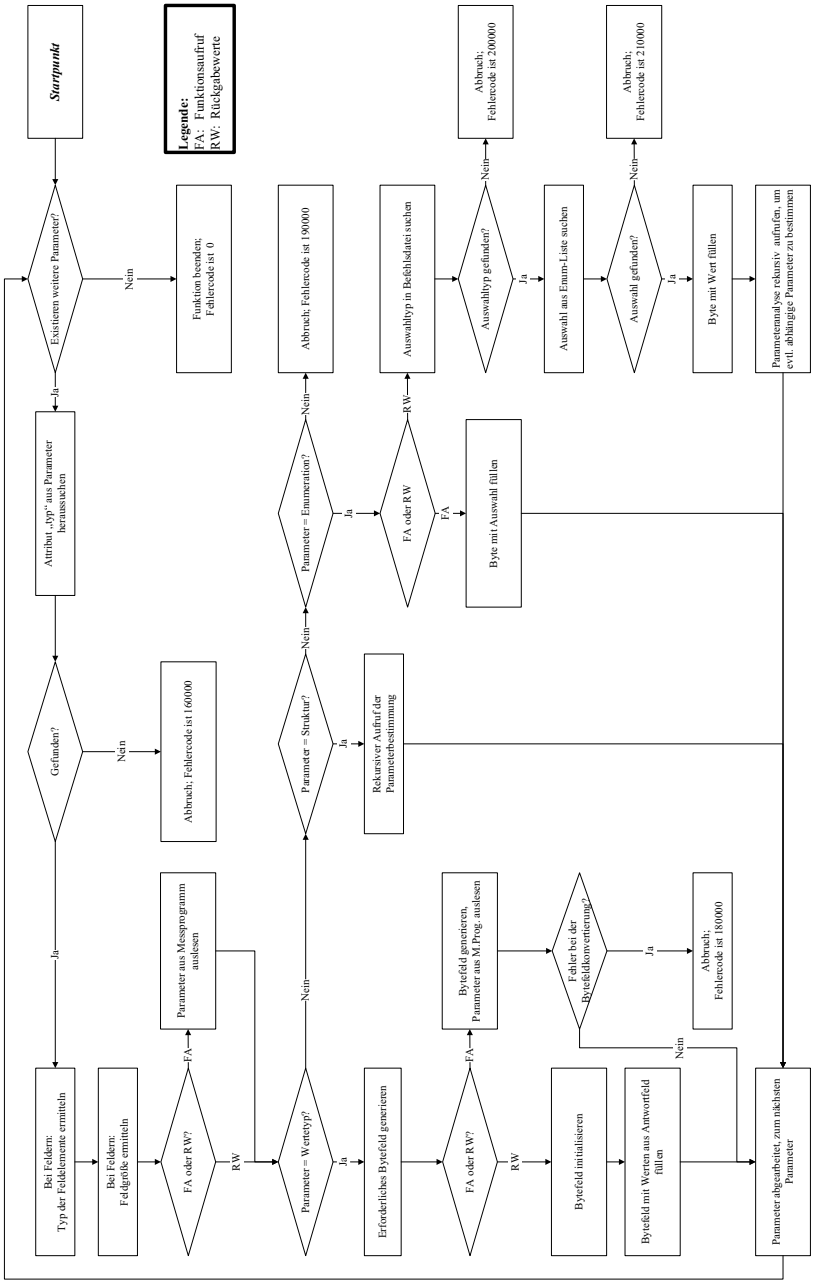


Bild 29: Bestimmung der Parameter, nach [19]

7 Teleservice

Durch die Realisierung des Teleservice wird es möglich, NPM-Maschine und Bedien-PC räumlich zu trennen. Damit werden einerseits messtechnische Vorteile erreicht – die Störungseinflüsse des Bedien-PC sind somit eliminiert – und andererseits kann die NPM-Maschine weltweit erreicht werden. Bei der Nutzung des Teleservice spielt es keine Rolle, ob die beiden Orte wenige Meter, mehrere Hundert oder sogar viele Tausend Kilometer voneinander entfernt sind. In jedem Fall muss die Steuerung, oder zumindest Teile davon, vom Bedien-PC der Maschine auf einen entfernten PC übertragen werden. In der Informationstechnik spricht man in diesem Fall von einer *Verteilten Anwendung*. Im Folgenden werden zunächst einige Technologien beschrieben, die es ermöglichen, verteilte Anwendungen zu erstellen. Des Weiteren wird in Kapitel 7.2 die Entwicklung einer Handsteuerung für die NPM-Maschine erläutert. Diese sollte dazu dienen, die Möglichkeit einer Fernsteuerung zu demonstrieren und die Unterschiede zwischen den beiden Technologien .NET Remoting und XML-Webdiensten darzustellen.

7.1 Verteilte Anwendungen

Um den Begriff *Verteilte Anwendung* zu erläutern, muss zunächst ein weiterer Begriff, der Begriff *Verteiltes System*, definiert werden. In [3] wird der Begriff wie folgt definiert:

> „Ein Verteiltes System ist definiert durch eine Menge von Funktionseinheiten oder Komponenten, die in Beziehung zueinander stehen (Client-Server-Beziehung) und eine Funktion erbringen, die nicht erbracht werden kann durch die Komponente alleine."

Tanenbaum leitet in [50] aus einer ähnlichen Definition zwei Aspekte ab. Der eine Aspekt bezieht sich auf die Hardware. Nach der genannten Definition ist ein verteiltes System auf mehrere autonome Rechner verteilt. Der zweite Aspekt, den Tanenbaum aus der Definition verteilter System ableitet, ist der Benutzer. Benutzer eines verteilten Systems haben demnach den Eindruck als würden sie es mit einem einzigen System zu tun haben. Das dahinter liegende System ist für den Benutzer völlig transparent. Die Anwendung scheint für den Nutzer lediglich aus dem Rechner zu bestehen, an dem er seine Eingaben tätigt.

Diese beiden Aspekte machen die Ziele verteilter Systeme deutlich:

- Zugang zu entfernten oder gemeinsam genutzten Ressourcen
  Im hier diskutierten, speziellen Fall der NPM-Maschine, können verteilte Systeme dazu beitragen entfernte Ressourcen zugänglich zu machen. Es ist dem Bediener somit möglich eine entfernt aufgestellte NPM-Maschine über Rechnergrenzen hinweg zu benutzen. Im Allgemeinen verfolgen verteilte Systeme auch das Ziel gemeinsame Informationsquellen zu nutzen. Dazu zählt beispielsweise der Zugriff auf gemeinsame Dateiverzeichnisse oder Datenbanken.

- Transparenz
  Für den Anwender soll ein verteiltes System transparent sein. Er soll im Idealfall nicht bemerken, dass die Anwendung, mit der er gerade arbeitet, auf mehreren Rechnern verteilt ausgeführt wird. Ein verteiltes System stellt sich also für den Bediener wie ein einziger Rechner dar.

- Offenheit
  Mithilfe geeigneter, zum Teil standardisierter, Schnittstellen kann ein verteiltes System seine Dienste anderen Anwendungen zur Verfügung stellen. Somit ist es möglich aus sehr unterschiedlichen Anwendungen heraus auf bereitgestellte Dienste zuzugreifen, vorausgesetzt, die jeweilige Anwendung implementiert die notwendigen Schnittstellen.

- Skalierbarkeit
  Verteilte Systeme sollen leicht erweiterbar sein. Dies betrifft sowohl das Hinzufügen weiterer Hardware als auch die Erweiterung des Funktionsumfangs angebotener Dienste.

Nachdem nun ein verteiltes System definiert und seine Ziele erläutert wurden, kann eine *Verteilte Anwendung* oder auch *Verteilte Applikation* als eine Software beschrieben werden, die aus mehreren Komponenten besteht. Die einzelnen Teile der gesamten Anwendung verteilen sich auf unterschiedliche Rechner. In Einzellfällen kann eine verteilte Anwendung, wenn auch in verschiedenen Prozessen, auch auf einem Rechner ausgeführt werden.

Neben der eigentlichen Anwendung, die auf mehrere Prozesse bzw. Rechner aufgeteilt wird, sind weitere Softwarekomponenten notwendig. Diese werden als *Middleware* bezeichnet. Die Middleware realisiert die Kommunikation der Anwendung mit dem darunterliegenden Betriebssystem des Rechners. In Bild 30 ist das entsprechende Modell zu sehen. Wie zu

erkennen ist, kann es durch geeignete Middleware realisiert werden die Anwendung auf Rechner mit unterschiedlichen Betriebssystemen zu verteilen.

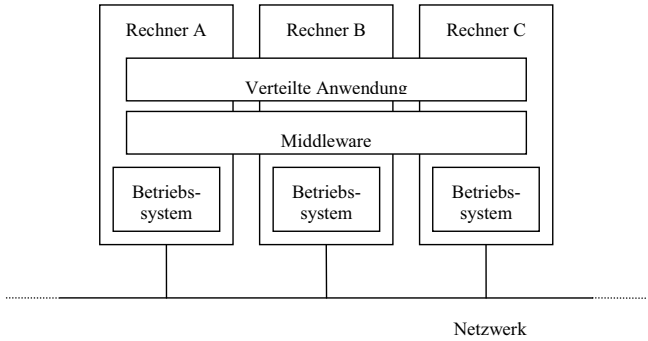


Bild 30: Verteilte Anwendung und Middleware, nach [50]

Das oben gezeigte Bild verdeutlicht die Schichten verteilter Systeme. Wie aber können die einzelnen Teile einer verteilten Applikation in eine logische Beziehung zueinander gebracht werden? Ein grundlegendes Modell zur Verdeutlichung ist das Client-Server-Modell. Beide Begriffe wurden bereits in der Definition des verteilten Systems angedeutet (siehe S. 71). Tanenbaum definiert beide Begriffe in [50] wie folgt:

> „Ein Server ist ein Prozess, der einen bestimmten Dienst implementiert, beispielsweise einen Dateisystemdienst oder einen Datenbankdienst. Ein Client ist ein Prozess, der einen Dienst von einem Server anfordert, indem er eine Anforderung sendet und dann auf die Antwort des Servers wartet."

Im Bild 31 ist eine allgemeine Client-Server-Struktur zu sehen. Der Client sendet zunächst eine Anforderung an den Server. Nach einer gewissen, durch den Übertragungsweg beeinflussten Verzögerung, erhält der Server die Nachricht, generiert die Antwort und sendet diese zurück an den Client. Nach einer erneuten Wartezeit erhält der Client die angeforderten Informationen und kann diese weiterverarbeiten.

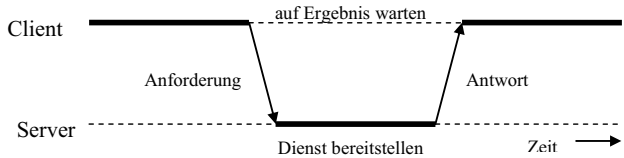


Bild 31: Zusammenarbeit zwischen Client und Server, Quelle: [50]

Diese vereinfachte Darstellung zeigt nur die prinzipielle Zusammenarbeit zwischen Client und Server. Nicht dargestellt sind die dahinter liegenden Prozesse, die das Senden einer Anforderung und das Empfangen einer Antwort oder die Speicherung von Daten ermöglichen. Softwaretechnisch gesehen werden die einzelnen Prozesse einer verteilten Anwendung verschiedenen *Schichten* zugeordnet. Meist wird der Begriff Schicht durch den englischen Begriff *Tier* ersetzt. In den meisten Quellen wird von einer Aufteilung auf drei Schichten gesprochen (*Three-Tier-Application*) [31], [28], [50], [3]. Diese sind:

- Präsentationsschicht (Client) Nutzerinterface, grafische Oberfläche
- Applikationsschicht (Server) Programmlogik, Zugriffe auf Daten
- Datenschicht (Server) Datenspeicherung

In einigen Quellen, besonders wenn es um die Programmierung von Applikationen für das Internet geht, wird eine weitere Schicht, eine sogenannte Webserver-Schicht, eingefügt. In diesem Fall wird auch von *Four-Tier-Application* gesprochen. Bild 32 zeigt die Schichten einer verteilten Anwendung. Die gestrichelt dargestellte Webserver-Schicht kann je nach Sichtweise mit der Schicht des Anwendungsservers zusammengefasst werden. Ebenfalls sind in Bild 32 bereits einige Technologien genannt, die eine verteilte Anwendung ermöglichen und zum Teil später beschrieben werden.

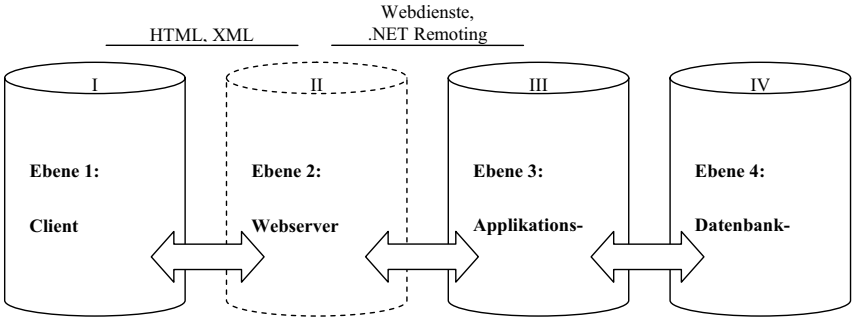


Bild 32: Schichtenmodell verteilter Anwendungen, nach [31]

Um eine verteilte Anwendung, in unserem Fall die Steuerung einer NPM-Maschine von unterschiedlichen Rechnern aus zu realisieren, ist, wie die vorangegangenen Abschnitte und Bilder zeigten, die Verwendung geeigneter Middleware-Technologien notwendig. In Bild 32 sind bereits einige Technologien aufgeführt. Es existieren allerdings eine ganze Reihe unterschiedlichster Technologien um verteilte Anwendungen zu realisieren. Im Rahmen

dieser Dissertation können nicht alle näher beschrieben werden. Um eine Einteilung vorzunehmen, werden die beschriebenen Technologien in zwei Gruppen aufgeteilt. Die eine Gruppe, nicht-browserbasierte Technologien (Kapitel 7.1.1) beschreibt zwei Technologien, die die reine Kommunikation zwischen zwei Programmteilen realisieren. Kapitel 7.1.2 beschreibt Technologien, bei den neben der eigentlichen Kommunikation zwischen den Programmprozessen auch die Art der Nutzerschnittstelle eine Rolle spielt. In diesem Fall handelt es sich um Bedienerschnittstellen in Form eines Internetbrowsers, z. B. dem Internet Explorer der Firma Microsoft. Die neuen Technologien der Firma Microsoft, zusammengefasst unter der Bezeichnung .NET, werden schließlich in Kapitel 7.1.3 beschrieben.

7.1.1 Nicht-Browserbasierte Technologien

Zum Bereich nicht-browserbasierte Technologien sollen diejenigen Middleware-Techniken zählen, bei denen der Client kein Browser ist. Entsprechende Nutzerschnittstellen der Ebene 1 (siehe Bild 32) können demnach Windows-Programme, Spezialsoftware oder auch Konsolenanwendung sein. Oftmals sind in diesem Bereich Technologien anzutreffen, die auf eigenen, zum Teil proprietären Protokollen basieren. Ebenfalls zu diesem Bereich können die später (Kapitel 7.1.3) beschriebenen .NET-Technologien von Microsoft gezählt werden. Da diese aber im Zusammenhang mit der vorliegenden Dissertation genauer untersucht und schließlich für die Realisierung verschiedener Komponenten verwendet wurden, wird diesen Techniken ein eigenes Kapitel gewidmet.

7.1.1.1 Remote Procedure Calls

Eine Anwendung stellt in der Regel eine Reihe an Funktionen zur Verfügung. In einer monolithischen Anwendung, d. h. einer nicht verteilten Anwendung, finden Prozedur- und Funktionsaufrufe lokal statt. Dazu wird der aufgerufenen Funktion eine Parameterliste übergeben, in deren Abhängigkeit dann das Ergebnis der Funktion berechnet wird. Der Rückgabewert der Funktion entspricht dann der Antwort, ist also das Ergebnis des Funktionsaufrufes.

Wird nun die Funktion durch einen entfernten Rechner zur Verfügung gestellt, spricht man von einem *entfernten Methodenaufruf*, einem *Remote Procedure Call* (RPC). Allerdings gibt es zwischen lokalen Funktionsaufrufen und RPCs entscheidende Unterschiede. Dazu muss zunächst einmal die Funktionsweise eines lokalen Methodenaufrufes erläutert werden. Bei einem lokalen Methodenaufruf befindet sich das gesamte Programm mit allen Methoden in einem gemeinsamen Adressbereich des Rechners. Die Parameter des Aufrufes werden auf

dem Stack der Anwendung abgelegt. Die Methode liest die Parameter und stellt nach Abarbeitung das Ergebnis wieder auf dem Stack zur Verfügung (Zur Vereinfachung wurde hier eine Unterscheidung in *call-by-value-* und *call-by-reference-Parameter* vernachlässigt! [50]). Im Falle eines RPCs existieren für den Client, dem Konsumenten der Funktion, und den Server, dem Anbieter der Funktion, jeweils ein eigener Adressbereich. Diese befinden sich in der Regel sogar auf unterschiedlichen Rechnern. Ein Zugriff auf einen gemeinsamen Speicherbereich, wie beispielsweise den Stack, ist nicht möglich. Deshalb werden entfernte Methodenaufrufe durch zwei spezielle Systemprozeduren durchgeführt. Diese Systemprozeduren heißen *Client-Stub* und *Server-Stub*. Der Client-Stub verpackt den Aufruf in eine Nachricht mit folgendem Inhalt:

- eine eindeutige Aufrufkennung
- die Adresse des Zielrechners (Servers)
- die Parameter in einem definierten Übertragungsformat

Der Server führt den Server-Stub in einer Endlosschleife aus. Trifft eine Nachricht ein, entpackt er die Parameter und ruft die entsprechende Methode auf. Aus Sicht des Servers handelt es sich in diesem Moment um einen lokalen Methodenaufruf. Nach Abarbeitung der Funktion verpackt der Server-Stub die Antwort wieder in eine Nachricht mit folgendem Inhalt:

- die eindeutige Aufrufkennung
- das Ergebnis des Funktionsaufrufes im definierten Übertragungsformat

Die Übertragung der Nachrichten übernehmen die Kommunikationsebenen der beiden Rechner. Auf diesen Mechanismus wird hier nicht eingegangen. Neben den von Windows her bekannten Remote Procedure Calls existiert eine Fülle weiterer RPC-Systeme. Drei bekannte Systeme sind die Systeme Xerox Courier RPC, Sun RPC und DCE RPC. Weitere Systeme finden sich in [3], [51] und [49].

Im Rahmen der Arbeiten an den Komponenten für die Steuerung von NPM-Maschinen wurden RPCs untersucht. Im Ergebnis dieser Untersuchung wurde entschieden, diese Technologie nicht einzusetzen.

7.1.1.2 TCP/IP-Sockets

Nahezu jedes Betriebssystem bietet eine Technologie für die synchrone Nachrichtenübertragung an: TCP/IP-Sockets. Eine Technologie, die als de facto-Standard das TCP/IP-Protokoll in Netzwerken nutzt [3].

Für die Kommunikation mittels Sockets muss das Betriebssystem sowohl auf der Server- als auch auf der Clientseite einen Kommunikationsendpunkt erzeugen. Weiterhin muss festgelegt werden, welches Protokoll für die Übertragung eingesetzt wird. Es existieren folgende Protokolle [3]:

- Datagram-Socket
  Ein verbindungsloses und damit unsicheres Protokoll für die Übertragung kurzer Nachrichten.

- Stream-Socket
  Ein verbindungsorientiertes bidirektionales Protokoll. Die Reihenfolge der Nachrichten wird sichergestellt.

- Raw-Socket
  Ein Protokoll, welches den Zugriff auf das Kommunikationsprotokoll ermöglicht.

- Packet-Socket
  Ein Protokoll, welches die Nachrichten in Pakete zerlegt. Auch hier ist die Reihenfolge der Nachrichten determiniert.

Die Technologie TCP-IP-Sockets wurde bereits 1981 an der University of California at Berkeley entwickelt. Im Vergleich mit anderen Technologien arbeiten Sockets auf einer sehr niedrigen Kommunikationsebene. Aufgrund der heute existierenden Technologien auf höheren Ebenen wurden in der NPM-Maschine keine TCP/IP-Sockets eingesetzt.

7.1.2 Browserbasierte Technologien

Der Zugriff auf entfernte Programmteile einer verteilten Anwendung kann auch durch browserbasierte Technologien erfolgen. Der Vorteil dabei liegt im Gegensatz zu den nicht-browserbasierten Technologien im Wegfall besonderer Software auf dem Clientrechner: Um auf Funktionen der entfernten Anwendungsteile zugreifen zu können, bedarf es lediglich eines einfachen HTML-Browsers, wie beispielsweise dem Internet Explorer von Microsoft. Somit wird der Zugriff auf serverseitig bereitgestellte Maschinendaten möglich, ohne dass

zusätzliche Software auf dem Client installiert werden müsste. Der Kontakt zu einer Maschine könnte so von jedem beliebigen Rechner mit Internetbrowser erfolgen, vorausgesetzt, ein direkt an der Maschine vorhandener Server stellt entsprechende Funktionen bereit. Im Folgenden sollen zwei Technologien erläutert werden, mit deren Hilfe derartige Anwendungen programmiert werden können.

7.1.2.1 Common Gateway Interface

Ein HTML-Browser dient dem Anzeigen von HTML-Dokumenten, die durch einen *Uniform Resource Locator* (URL) eindeutig identifiziert und aufgerufen werden können. Der Inhalt einer auf einem Server gespeicherten HTML-Seite ist aber in der Regel statisch. Für die Realisierung von Teleservicefunktionen muss jedoch ein aktueller Zustand der Maschine oder sonstiger Daten angezeigt werden können. Dazu muss ein Programm auf dem Server die gewünschten Daten ermitteln und eine HTML-Seite mit aktuellen Werten generieren. Der Server sendet diese Seite als Antwort zurück zum Client.

Das Common Gateway Interface, abgekürzt CGI, ist ein Standard, der den Datenaustausch zwischen einem Web-Server und Programmen auf diesem Server ermöglicht. Die Programme werden dazu von HTML-Seiten aufgerufen und vom Server gestartet. Mit den an das Programm übergebenen Parametern werden die gewünschten Informationen ermittelt und in Form einer neu generierten HTML-Seite zurück an den Client gesendet.

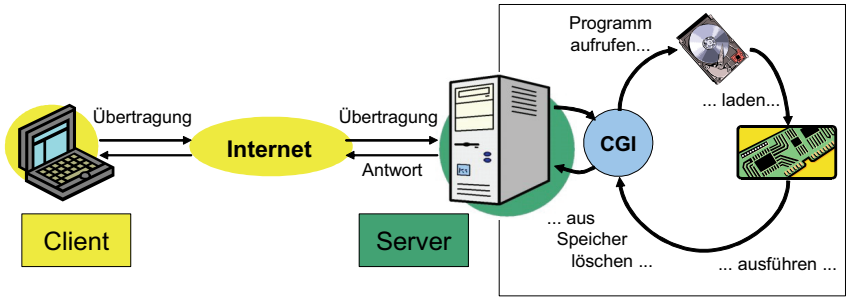


Bild 33: Informationsfluss beim CGI [13]

Zunächst ruft der Client über einen URL eine statische HTML-Seite auf. Der Inhalt dieser Seite umfasst in der Regel ein sogenanntes HTML-Formular. Das sind spezielle HTML-Elemente, die es dem Nutzer ermöglichen, Daten einzugeben. So gibt es beispielsweise Formularelemente für Text- oder Passworteingaben, Checkboxen und Radiobuttons. Mithilfe zusätzlicher Schaltflächen können die Daten des Formulars gelöscht oder an den Server

übertragen werden. Der folgende Quellcode eines Formulars stellt zwei Textfelder, zwei Optionsfelder und Schaltflächen für das Senden und Rücksetzen des Formulars bereit.

```
<form method="GET" action="http://localhost/CGI/Rechner.cgi">
    <input name="X" type="text" value="2" />
    <input name="Y" type="text" value="2" />
    <input checked="checked" name="Art" type="radio" value="addition" />
    <input name="Art" type="radio" value="subtraktion" />
    <input type="submit" value="Berechnen" />
    <input type="reset" value="Zurücksetzen" />
</form>
```

Beim Drücken der Schaltfläche vom Typ *Submit* werden die Daten an den im Attribut *Action* angegebenen URL gesendet. Der URL zeigt dabei auf das auszuführende Programm. Die Serversoftware muss nun in der Lage sein, den Inhalt der Formulareingaben auszulesen und das aufgerufene Programm zu starten. Dazu werden auf dem Server spezielle Umgebungsvariablen gesetzt, um dem CGI-Programm Informationen über den Aufruf bereitzustellen. Dazu zählen unter anderem Informationen über den aufrufenden Client, die Anzahl der übertragenen Daten und schließlich die Daten selbst.

Durch das Attribut *Method* des Form-Tags wird festgelegt, wie der Server die Parameter an das Programm übergeben soll. Die Methode GET codiert dazu die Formulareingaben und fügt diese, angeführt von einem '?', an den URL des auszuführenden Programms an:

```
http://localhost/CGI/Rechner.cgi?X=7&Y=3&Art=addition
```

Der Server stellt anschließend die codierten Daten in der Umgebungsvariable *QUERY_STRING* bereit. Das CGI-Programm kann direkt auf den Inhalt dieser Variable zugreifen und die eingegebenen Formulardaten auswerten. Im Unterschied zu GET wird durch die Methode POST ein Eingabekanal des Servers verwendet. Dazu ermittelt der Server den Umfang der übertragenen Datenmenge und speichert die Anzahl übertragener Bytes in der Umgebungsvariable *CONTENT_LENGTH* ab. Das CGI-Programm muss entsprechend viele Zeichen über den Eingabekanal einlesen. Die eingelesene Zeichenkette beinhaltet die Eingaben des Formulars und kann entsprechend ausgewertet werden. Der codierte Inhalt des Formulars wird also entweder direkt über eine Umgebungsvariable (Methode GET) oder über einen Eingabe-Kanal (Methode POST) an das CGI-Programm übergeben. Eine codierte Zeichenfolge könnte in beiden Fällen beispielsweise wie folgt aussehen:

```
X=7&Y=3&Art=addition
```

Mit den aus dieser Zeichenkette gewonnenen Parametern kann das CGI-Programm die gewünschten Informationen bereitstellen und über Standardausgaben ein vollständiges HTML-Dokument in einen Ausgabekanal des Servers schreiben. Dieser sendet diese dynamisch erzeugte HTML-Seite zurück an den Client.

Ein Vorteil bei der Verwendung des Common Gateway Interfaces ist die Möglichkeit, CGI-Programme oder -Skripte in nahezu jeder beliebigen Sprache erstellen zu können. Es muss lediglich ein entsprechender Interpreter oder eine entsprechende Laufzeitumgebung auf dem Server vorhanden sein. Gleichzeitig lassen sich die Programme leicht testen, da sich die Programme bei Vorgabe einer Testzeichenkette auch ohne das Vorhandensein von Serversoftware ausführen lassen.

Nachteilig wirken sich bei der Verwendung des Common Gateway Interfaces Geschwindigkeits- und Sicherheitsaspekte aus. Die Performance des Servers wird durch den Einsatz von CGI stark beeinträchtigt, weil für jeden CGI-Aufruf das Programm erneut in den Speicher des Servers geladen und gestartet werden muss. Bei mehreren Anfragen liegen mehrere Kopien des Programms gleichzeitig im Speicher. Andere Ansätze wie FastCGI versuchten diesen Nachteil zu kompensieren, konnten sich aber nicht gegen die sich gleichzeitig entwickelnden Technologien wie ASP durchsetzen. Sicherheitstechnisch gesehen stellen CGI-Programme und -Skripte ein Risiko dar, da der Server das Ausführen von Programmen von außen gestatten muss.

Das Common Gateway Interface wird heute hauptsächlich aufgrund seiner schlechteren Performance gegenüber anderen, moderneren Browser-basierten Technologien kaum noch verwendet. Beispielsweise hat sich ASP.NET – siehe Kapitel 7.1.3, Microsoft .NET Technologien – durchgesetzt.

7.1.2.2 Active Server Pages

Bereits das Common Gateway Interface bot eine Schnittstelle, um bestimmte Aktionen auf einem Server ausführen zu lassen. Bei CGI bestanden diese Aktionen darin, ein Programm zu starten, diesem Parameter zu übergeben und die vom Programm erzeugte HTML-Seite an den Client zurückzusenden. Eine weitere Technik den Inhalt einer HTML-Seite auf dem Server verändern zu lassen bieten die sogenannten *Server Side Includes* (SSI). Sie stellen eine Erweiterung der Serverfunktionalität dar. Im Fall des von Microsoft entwickelten Webservers, dem *Internet Information Services* (IIS), heißt die Programmierstelle mit deren Hilfe auf die Erweiterungen zugriffen werden kann *Internet Server Application Programming Interface*

(ISAPI). Genutzt wird diese Schnittstelle durch die Technologie Active Server Pages. *Active Server Pages* (ASP) sind Dateien mit der Dateinamenerweiterung *.asp*. Sie beinhalten neben HTML-Code auch Skripte. Durch die besondere Dateiendung werden die Dateien bei Aufruf auf dem Server verarbeitet. Die eingebetteten Skripte greifen über die ISAPI-Schnittstelle auf entsprechende Serverfunktionen zurück. Der HTML-Code der aufgerufenen Seite wird entsprechend verändert und an den Client gesendet. Die Standard-Skriptsprache ist eine Version von Visual Basic – *VBScript*. Es sind aber auch andere Sprachen möglich, so beispielsweise JScript, PearlScript, HaskellScript usw.

Der folgende Quellcode zeigt eine sehr einfache ASP-Seite. Ruft der Client über einen Browser den URL der Datei auf (z. B. http://<servername>/Zeit.asp), wird das eingebettete Skript auf dem Server verarbeitet, der entsprechende Wert eingesetzt und als HTML-Seite an den Client geschickt.

```
<%@ Language = "VBScript" %>
<html>
      <body bgcolor=white>
            <Script RUNAT="Server" language="VBScript">
                  Response.Write "Es ist jetzt "& time & " Uhr!<br>"
            </Script>
      </body>
</html>
```

Mit Hilfe von Unterfunktionen können die Skripte strukturiert werden. Durch Veränderung des Ausgabeformates können mithilfe von ASP-Skripten sogar Grafiken erzeugt werden. Weiterhin kann ASP auf COM-Objekte zugreifen. Die wohl am häufigsten eingesetzten COM-Klassen sind die Klassen der Komponente *ActiveX Data Object* (ADO) [42]. Mit diesen Klassen kann der Server auf Datenbanken zugreifen und entsprechende Abfragen durchführen.

Die Parameterübergabe an das serverseitige Skript erfolgt ähnlich wie bei CGI über Querystrings. Auch hier ist die Verwendung von GET und POST möglich. Weiterhin bietet ASP eine auf Cookies basierende Sitzungsverwaltung an. Damit ist es den Programmierern von ASP-Seiten ermöglicht worden, sitzungsbezogene Informationen in einer *Session-Collection* zu speichern. Dies kann mit einer Art serverseitigen, globalen Variable verglichen werden. Der *Sitzungs-Cookie* wird auf dem Client gespeichert und kann bei Bedarf an den Server gesendet werden. Damit kann der Server eine gespeicherte Sitzung rekonstruieren und somit eventuell bereits ermittelte Daten darstellen. Die ist beispielsweise bei der Programmierung eines Web-Shops sehr hilfreich.

ASP bietet im Vergleich zu CGI viele Vorteile in der Entwicklung browserbasierter, verteilter Anwendungen. Mit der Einführung der .NET-Technologien im Jahr 2002 ist jedoch die Weiterentwicklung der Active Server Pages eingestellt worden. Als Ersatz für ASP führte Microsoft die heute verwendete Technologie ASP.NET ein. Sie ist also direkter Nachfolger und Ersatz für ASP. Im Rahmen der Dissertation wurden deshalb keine weiteren Untersuchungen zu ASP durchgeführt. ASP.NET bildet jedoch die Grundlage für Webdienste, die mit dem .NET Framework entwickelt werden. Diese Technologie wird im folgenden Kapitel näher beschrieben.

7.1.3 Microsoft .NET Technologien

Im Juni 2000 sprach Bill Gates auf mehreren Tagungen über eine Initiative von Microsoft, sich von der traditionellen Softwareentwicklung zu verabschieden und das Internet zu einer dienstleistungsorientierten Softwareplattform auszubauen [39]. In einer am 22. Juni gehaltenen Rede auf dem Forum 2000 in Redmond hieß es [16]:

> „...verkündet Microsoft heute, dass unsere Bemühungen als Firma um die Erzeugung einer neuartigen Plattform fokussiert werden. Wir nennen sie .NET. Das ist eine Bezeichnung, von der Sie heute noch viel hören werden. Aber dahinter steht mehr, als man zunächst denkt. .NET steht für die Idee umfangreichen Code auf allerart [Anm.: in Hinblick auf Hard- und Software unterschiedliche) Clients auszuführen. .NET steht für die Idee des Zugriffs auf (Web-)Services über das Internet, um jeder Art von Client Dienste anbieten zu können. Und schließlich steht es für eine neue Generation von Servern, die untereinander kommunizieren und ihre Dienste bereitstellen...“

Man kann .NET als eine Art Markenname verstehen, der von Microsoft für mehrere unterschiedliche Technologien verwendet wird. Kerntechnologie der .NET-Technologien sind die Webdienste (auch XML-Webdienste). Webdienste stellen Funktionen bereit, die über das Internet aufgerufen werden können. Mehr dazu aber im entsprechenden Abschnitt. Neben den Webdiensten führt Microsoft auch folgende Technologien unter der Bezeichnung .NET:

- .NET-Framework
- Visual Studio .NET: Die neue, auf .NET zugeschnittene Entwicklungsumgebung.

Im Jahr 2001 zählten auch noch die *.NET My Services (Hailstorm)*, eine Sammlung kostenpflichtiger und kostenloser, vordefinierter Webdienste und die *.NET Enterprise Server*, eine Familie von Software-Servern zur „Vision“ von .NET. Diese Technologien werden aber

heute von Microsoft nicht mehr in Zusammenhang mit .NET genannt oder bestehen nicht mehr in der ursprünglichen Form [8]. Die endgültige Veröffentlichung der angekündigten Technologien und Komponenten erfolgte dann im Januar 2002 [42].

7.1.3.1 Das .NET Framework

Das *Component Object Model* (COM) war eine von Microsoft entwickelte Plattformtechnologie für die objektorientierte Programmierung unter Windows. COM-Objekte stellen über eine Schnittstelle ihre Funktionen bereit. Die Verbreitung dieser und verwandter Technologien wie DCOM oder COM+ war bis zur Einführung der .NET-Technologien enorm. Nahezu jede Windows-Anwendung basierte auf diesen Technologien. Dementsprechend groß waren die Veränderungen die .NET mit sich brachte [8].

Das .NET-Framework ist für Entwickler der wichtigste Teil von .NET. Das .NET Framework ist eine Plattform zur Entwicklung und Ausführung von Programmen. Das .NET Framework selbst besteht wiederum aus zwei Teilen: Der *Common Language Runtime* (CLR) und einer umfangreichen Sammlung an Funktionen und Klassen, der *.NET Framework Klassenbibliothek*. Bild 34 zeigt den Aufbau des .NET Framework und seine Beziehung zum darunter liegenden Betriebssystem und möglichen, auf dem Framework aufbauenden, Anwendungen.

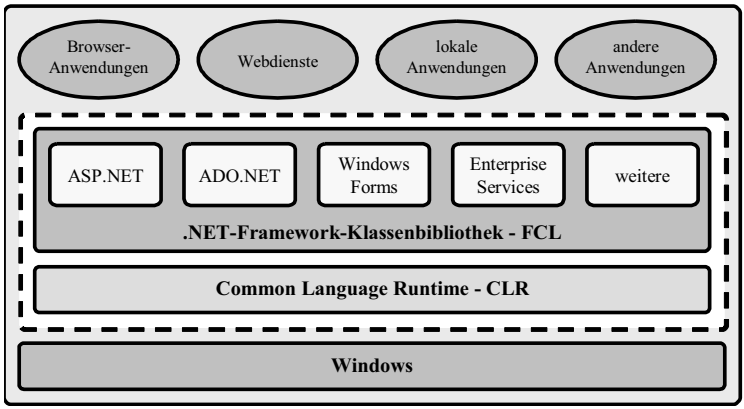


Bild 34: Aufbau des .NET Framework (gestrichelter Rahmen) [8]

Die CLR soll für die Programmierer eine Basis von Datentypen und Diensten bereitstellen. Dabei ist diese Basis unabhängig von der verwendeten Programmiersprache. Trotzdem hat Microsoft die Programmiersprachen Visual Basic .NET und C# bereitgestellt, die speziell auf

die CLR abgestimmt sind. Quellcode, der auf der CLR aufbaut, wird als *verwalteter Code* bezeichnet.

Im Unterschied zur herkömmlichen Programmerstellung wird der auf der CLR basierende Quellcode nicht in prozessorspezifischen Maschinencode sondern zunächst in eine prozessorunabhängige Zwischensprache, die *Microsoft Intermediate Language* (MSIL) kompiliert. Erst zur Laufzeit wird dieser Zwischencode mithilfe des *Just in Time Compilers* (JIT-Compiler) in nativen Code übersetzt. Diese Vorgehensweise hat den Vorteil, dass die Dateien bis zur Verwendung maschinenunabhängig sind und der JIT-Compiler prozessorspezifische Optimierungen vornehmen kann.

Die .NET Framework-Klassenbibliothek (FCL) stellt eine Sammlung von Klassen und Typen zur Verfügung. Mehrere tausend Klassen sind in einer hierarchischen Struktur zu einem Baum zusammengefügt. Die Wurzel der Klassenbibliothek bildet der Namensraum *System* mit seinen grundlegenden Datentypen. Davon leitet sich eine Vielzahl unterschiedlicher Namensräume für unterschiedlichste Einsatzgebiete ab. Um nur einige wenige Beispiele zu nennen, sei an dieser Stelle auf die Namensräume *System.Web*, *System.Windows.Forms* und *System.Xml* hingewiesen.

Seit Januar 2002 hat das .NET Framework zahlreiche Überarbeitungen erfahren. Die erste Version, das .NET Framework 1.0, wurde im Jahr 2003 mit der Einführung des Visual Studio .NET 2003 durch das .NET Framework 1.1 ersetzt. Ende 2005 wurde, wieder zusammen mit der neuen Entwicklungsumgebung Visual Studio 2005, die Version 2.0 veröffentlicht und seit Ende 2006 existiert die nur leicht erweiterte Version 3.0 des .NET Frameworks. Für den wachsenden Markt der mobilen PCs und Anwendungsgeräte existiert zusätzlich eine im Funktionsumfang reduzierte Version des .NET Frameworks: Das *.NET Compact Framework*.

7.1.3.2 .NET Remoting

In der objektorientierten Programmierung werden Funktionen oder Methoden in sogenannten Klassen zusammengefasst. In konventionellen Programmen sind diese Methoden nur Programmteilen zugänglich, die in derselben Anwendungsdomäne betrieben werden [39]. Wird diese Klasse innerhalb einer verteilten Anwendung auch anderen zur Verfügung gestellt, wird sie als remotefähige Klasse bezeichnet. Das .NET-Framework bietet dafür eine spezielle Klasse an, von der remotefähige Klassen abgeleitet werden können: Die Klasse *System.MarshalByRefOject*. Solche, von MarshalByRefObject abgeleitete Klassen, können sowohl in der Anwendungsdomäne des Servers, als auch in der des Clients instanziiert

werden. Das heißt, der Client kann die Methoden der remotefähigen Klasse so benutzen, als würden sie sich in seiner eigenen Programmumgebung befinden.

Wie in Bild 35 zu sehen ist, handelt es sich bei der clientseitigen Instanz der remotefähigen Klasse um ein Abbild des eigentlichen Objekts, einem sogenannten *Proxy*. Der Proxy hat dabei das gleiche Erscheinungsbild wie die Klasse auf dem Server, sie dient aber lediglich als Verweis auf das serverseitige Objekt. Der Zugriff auf die Funktionen der Proxy-Klasse erfolgt für den Programmierer vollkommen transparent.

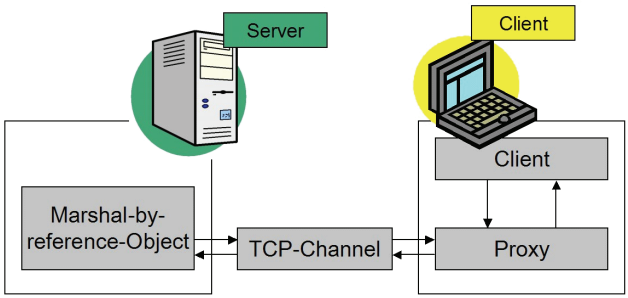


Bild 35: Struktur einer .NET Remoting Anwendung, [55]

Für die Kommunikation zwischen der Proxy-Klasse auf dem Client und dem Remoteobjekt auf der Serverseite dient ein Kanal. Innerhalb des .NET-Frameworks existieren hierfür zwei Varianten. Das ist zum einen der tcp-Kanal auf der Basis des TCP/IP-Protokolls und zum anderen der http-Kanal, der auf dem http-Protokoll beruht. Aufgrund des schlankeren Protokolls des tcp-Kanals wird er als der Leistungsfähigere eingeschätzt [39]. Die Wahl des Kanals ist dem Programmierer freigestellt. Soll aber das Remoteobjekt vom IIS verwaltet werden, ist die Verwendung des http-Kanals notwendig.

Vorteil von .NET Remoting ist die Integration in das .NET Framework und die damit verbundene gute Anbindung an Programme, die auf diesem aufbauen. Weiterhin stellt die Verbindung auf Basis des tcp-Protokolls eine schlanke, schnelle Übertragung von Nachrichten und Parametern innerhalb der verteilten Anwendung sicher.

Die Verwendung eines tcp-Kanals unter Angabe eines Ports kann aber auch zu einem Nachteil von .NET Remoting werden: Die Verbindung kann an Rechnergrenzen durch Firewalls blockiert werden. Eine Verbindung ist dann nicht oder nur mit hohem Administrationsaufwand möglich. Ein weiterer Nachteil ist die Tatsache, dass .NET Remoting nur innerhalb des .NET Framework lauffähig ist. Eine Plattformunabhängigkeit ist damit nicht gewährleistet. Client und Server müssen das .NET Framework implementieren.

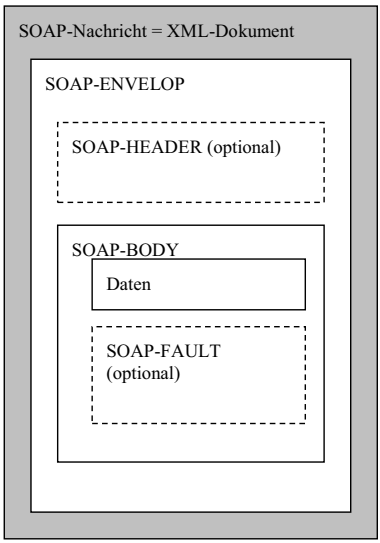
Eine nähere Beschreibung der Funktionsweise von .NET Remoting erfolgt an einem konkreten Beispiel im Kapitel 7.2.1 in Zusammenhang mit der Vorstellung einer auf .NET Remoting basierenden Handsteuerung für eine NPM-Maschine.

7.1.3.3 XML-Webdienste

XML-Webdienste, oft auch nur Webdienste oder Webservices genannt, sind eine weitere Technologie zur Realisierung verteilter Anwendungen. Diese besitzen keine Benutzeroberfläche, sondern stellen ihre Dienste in Form von *Webmethoden* im Internet bereit. Die Webmethoden werden vom Client der Anwendung mittels http-Anfragen aufgerufen. Die Antwort des Servers ist nicht wie bei ASP eine HTML-Seite, sondern eine Nachricht in Form von XML-Daten. Daher wird auch der Begriff *XML-Webdienste* benutzt. Häufig werden für XML-Webdienste auch synonym die Begriffe Webdienste und Webservices (oder eine andere Schreibweise: Web Services) verwendet.

Da Webdienste keine Oberfläche besitzen, können die für den Zugriff auf die Webmethoden erstellten Clients neben Windows-Programmen, Webseiten oder Konsolenprogrammen auch weitere beliebige Anwendungen eingesetzt werden, die XML-Daten auswerten können. XML-Webdienste bieten also eine ideale Möglichkeit, um plattformunabhängige verteilte Anwendungen zu schreiben. Sie entsprechen in Ihrer Struktur einer idealen Middleware-Technologie nach Bild 32.

Die zwischen Client und Server übertragenen Daten werden nach einem XML-basiertem Protokoll übertragen. Dieses Protokoll ist das herstellerunabhängige *Simple Object Access Protocol* (SOAP). SOAP strukturiert den Datentransport nach folgendem Schema:




```
<?xml version="1.0" encoding="utf-8" ?>

<soap:Envelope
       xmlns:soap="http://schemas.xmlsoap.org/
                                    soap/envelope">
      <soap:Header>
            <user>username</user>
            <password>geheim</password>
      </soap:Header>

      <soap:Body>

            <CountResponse>1</CountResponse>

            <soap:fault>
                  <faultstring>
                        Fehlermeldung
                  </faultstring>
            </soap:fault>
```

Bild 36: Struktur und Beispiel für eine SOAP-Nachricht, (Beispiel nach [30])

Eine SOAP-Nachricht liefert also einen *Umschlag*, der die Daten in ein vorgeschriebenes Schema verpackt. Der genauere Aufbau des SOAP-Protokolls kann beispielsweise in [59] nachgelesen werden.

Webdienste basieren aus Sicht des .NET Framework auf ASP.NET. Bei der Entwicklung eines Webdienstes mit dem Visual Studio heißt der auszuwählende Projekttyp daher auch *ASP.NET Web Service*. Wie bereits angesprochen ist ASP.NET der im .NET Framework integrierte Nachfolger von ASP. Zwei wichtige Unterschiede zwischen ASP und ASP.NET in Bezug auf Webdienste sind folgende:

- ASP.NET Anwendungstypen

  ASP war nur für die Erstellung von Webformularen gedacht. ASP.NET hingegen stellt einen weiteren Anwendungstyp bereit: Webdienste. Beide Anwendungstypen unterscheiden sich unter anderem in der verwendeten Dateinamenserweiterung. Webformulare (Webforms) besitzen die Erweiterung *.aspx*, Webdienste hingegen die Erweiterung *.asmx*. Anhand dieser Unterscheidung entscheidet der IIS über die Verarbeitung der jeweiligen Datei.

- Code-Behind-Programmierung

  Bei ASP.NET wird der Quellcode der Anwendung in einer separaten Datei ausgelagert. Anschließend wird der Quellcode in MSIL kompiliert und liegt anschließend als DLL

vor. Bei der Kompilierung erfolgt sofort eine Syntax-Prüfung, was die Fehlererkennung schon während der Programmierung, nicht erst zur Laufzeit ermöglicht. Weiterhin ist die Verarbeitung des in der Zwischensprache vorliegenden Codes durch den JIT-Compiler schneller als die Interpretation von ASP-Seiten.

Der Client sendet seine Anforderung mittels XML-Daten an den Server. Dies erfolgt durch einen GET- oder POST-Befehl, der den URL der gewünschten asmx-Datei beinhaltet. Der Server untersucht die aufgerufene Datei. Diese muss die *Direktive WebService* und einen Verweis auf die Klasse mit den Webmethoden enthalten.

```
<%@ WebService Language="c#" Codebehind="serv.asmx.cs" Class="service1" %>
```

Die Methoden können nun entsprechend abgearbeitet werden. Anschließend werden die Ergebnisse wieder in eine SOAP-formatierte XML-Nachricht verpackt und an den Client gesendet. Die Syntax des Quellcodes eines Webdienstes wird ausführlicher im Kapitel 7.2.2 beschrieben.

Zum Ausführen von Webdiensten müssen mehrere Dateien vom IIS verwaltet werden. Dieser muss zunächst ein virtuelles Verzeichnis anlegen. Innerhalb des Verzeichnisses werden die asmx-Datei, der dazugehörige Quellcode sowie zwei weitere Dateien zur Steuerung der Anwendung erzeugt. Weiterhin existiert das Unterverzeichnis *bin*, das den kompilierten Zwischencode in Form einer DLL enthält. Bild 37 zeigt ein Beispiel für eine Verzeichnisstruktur innerhalb der Internet Information Services. Der gezeigte Beispielwebdienst würde standardmäßig über den URL *http://<serverName>/NPM-Webdienst/NPM-Service.asmx* aufgerufen werden können.

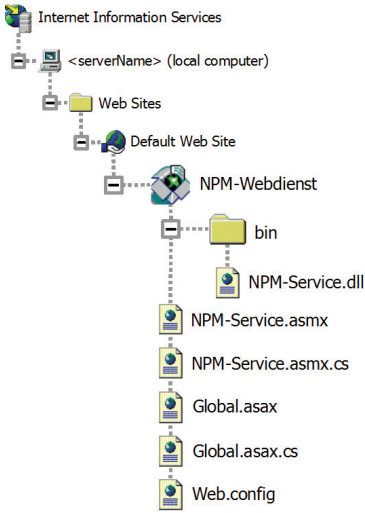


Bild 37: Verzeichnisstruktur im IIS

Die asmx-Datei bildet wie erwähnt den Programmeinstiegspunkt. Mithilfe der Datei Global.asax und der dazugehörigen Quellcodedatei Global.asax.cs können bestimmte Ereignisse des Webdienstes gesteuert werden. So kann der IIS beispielsweise beim Starten des Dienstes weitere Programme aufrufen. Die xml-basierte Datei Web.config steuert weitere Eigenschaften der Anwendung. Hier kann beispielsweise der Authentifizierungsmechanismus festgelegt werden, mit dem sich ein Client gegenüber dem Server identifizieren muss.

Der IIS übernimmt auch die Schnittstellenbeschreibung des Webdienstes. Diese Schnittstelle wird allen Clients zur Verfügung gestellt, um zu prüfen, welchen Funktionsumfang der Webdienst besitzt. Die Beschreibung der Schnittstelle erfolgt durch die auf xml-basierende Sprache *Web Service Description Language* (WSDL). Um diesen *Schnittstellenvertrag* für einen konkreten Webdienst einzusehen, wird dieser mit dem Parameter *?wsdl* aufgerufen:

```
http://<serverName>/NPM-Webdienst/NPM-Service.asmx?wsdl
```

Die Antwort des IIS ist eine XML-Datei, die alle Informationen über Methoden, Parameter und Rückgabewerte enthält. Weitere Informationen über die WSDL sind unter [57] zu finden.

XML-Webdienste sind aufgrund ihrer vielen Vorteile hinsichtlich Plattformunabhängigkeit, Transparenz und nicht zuletzt wegen der umfangreichen Unterstützung durch das Visual Studio, zu einer der wichtigsten Technologien für verteilte Anwendungen geworden. Im Rahmen dieser Dissertation sind deshalb mehrere Komponenten zur Steuerung von NPM-

Maschinen durch XML-Webdienste erweitert worden. Das folgende Kapitel zeigt ein Beispiel für ein Programm, welches mithilfe eines Webdienstes die Steuerung einer NPM-Maschine über das Internet hinweg ermöglicht.

7.2 Prototyp einer fernbedienbaren Handsteuerung

Über entsprechende Befehle des in den Kapiteln 2.2 und 5.3 beschriebenen Befehlssatzes der NPM-Maschine, kann der Messtisch bewegt werden. Dies ist sowohl während der Abarbeitung eines Messprogramms als auch im besonderen Maße während der Einrichtungsphase möglich und notwendig. Ein konkretes Beispiel wäre die Einrichtung der Maschine für einen Messvorgang. Dabei ist zunächst das Messobjekt auf dem Messtisch der NPM-Maschine zu befestigen. Um nun einen Bezug zwischen Messsystem und -objekt herstellen zu können, muss das Messobjekt angetastet werden. Dazu wird der Messtisch in kleinen Schritten positioniert, bis das Messsystem einen Kontakt zur Oberfläche des Prüflings anzeigt. Um diesen Vorgang zu unterstützen, wurde im Rahmen dieser Dissertation ein entsprechendes Windows-Programm geschrieben. Dieses ermittelt zunächst die angeschlossenen Geräte und listet sie in einem Auswahlfeld auf. Nach Auswahl der gewünschten Maschine kann der Nutzer die Maschine aktivieren. Aktivieren bedeutet, dass der Messtisch in die Nullposition der Antriebe bewegt und somit der Maschinennullpunkt festgelegt wird. Anschließend kann über die Eingabe von Koordinaten der Messtisch an frei definierbare Raumpunkte gefahren werden. Mit dem Steuerelement *Schrittbetrieb* kann der Tisch schrittweise in allen Raumachsen bewegt werden. Der Schrittbetrieb lässt sich per Maus bedienen. Es existiert aber auch eine Schnittstelle, die es ermöglicht, spezielle Eingabegeräte, wie z. B. einen Spaceball, einzubinden. In Bild 38 ist die Oberfläche des Programms zu sehen.

Das Programm wird durch ein zusätzliches Fenster erweitert, dass schematisch die Position des Messtisches im Raum anzeigt. Die dazu erforderlichen Koordinaten werden in Echtzeit aus der Maschine gelesen und visualisiert. Aus Gründen der Übersichtlichkeit ist dieses Programmfenster aber nur im Anhang abgebildet.

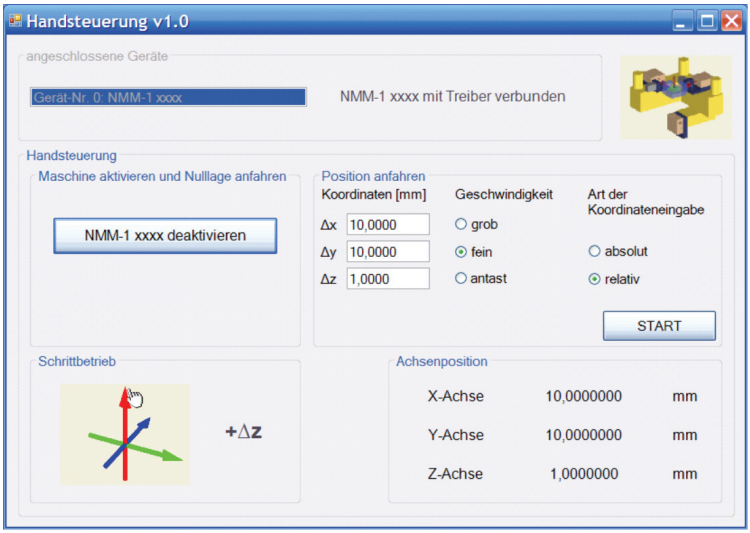


Bild 38: Bedienoberfläche der Handsteuerung

Das Programm wurde in der Programmiersprache C# erstellt und so erweitert, dass es Teleservicefunktionalitäten demonstriert. Die Erweiterung führte also zu einer verteilten Anwendung. Die Steuerung des Messtisches sollte von einem entfernten Rechner aus erfolgen können. Der Funktionsnachweis wurde für die Technologien .NET Remoting und Webdienste erbracht. Die folgenden Abschnitte beschreiben die Realisierung beider Verfahren.

7.2.1 Realisierung mit .NET Remoting

.NET Remoting beruht, ähnlich wie RPC (Kapitel 7.1.1.1) auf der Veröffentlichung von Methoden einer Klasse. Der Methodenaufruf kann dadurch anderen, auch entfernten Clients zur Verfügung gestellt werden. Für das Beispielprogramm *Handsteuerung* wurde zunächst eine lokale Klasse erstellt, die verschiedene Methoden für die Kommunikation mit der NPM-Maschine bereitstellt. Die Programmierung erfolgte in der Sprache C# mithilfe des Visual Studio 2005. Der Rumpf der Klasse:

```
public class NPMRemoteClass
{ ... }
```

Die durch die Klasse bereitgestellten öffentlichen Methoden ermöglichen es, den Zustand der Maschine auszulesen und den Messtisch zu positionieren. Im Einzelnen sind das folgende Methoden:

- `public int CountDevices(){ ... }`
  Liefert die Anzahl der an den Bedien-PC angeschlossenen NPM-Maschinen.

- `public string DeviceSerialNumber(int DeviceNumber){ ... }`
  Ermittelt eine im USB-Modul der NPM-Maschine gespeicherte Kennung des Gerätes.

- `public bool ConfigDevice(int DeviceNumber){ ... }`
  Konfiguriert die Verbindung zwischen Treiber und einem durch den Parameter *DeviceNumber* angegebenen Gerät.

- `public void UnConfigDevice(int DeviceNumber){ ... }`
  Trennt die Verbindung zwischen Treiber und dem angegebenen Gerät.

- `public int Activate(int DeviceNumber){ ... }`
  Aktiviert die angegebene NPM-Maschine. Dadurch werden die Achsen in die Nullposition gefahren.

- `public int DeActivate(int DeviceNumber){ ... }`
  Deaktiviert die angegebene NPM-Maschine.

- `public bool GetActivationState(int DeviceNumber){ ... }`
  Liefert einen booleschen Wert der angibt, ob die angegebene NPM-Maschine bereits aktiviert ist.

- `public bool GetMotionState(int DeviceNumber){ ... }`
  Liefert einen booleschen Wert der angibt, ob sich die Achsen der angegebenen NPM-Maschine in Bewegung befinden.

- ```
  public int MoveToPoint(int DeviceNumber, double X, double Y, double Z,
                         int mode, int MoveSpeed){ ... }
  ```
  Fährt den Messtisch der angegebenen Maschine an den durch *X*, *Y* und *Z* definierten Punkt. Die Koordinatenangabe kann durch den Parameter *mode* relativ oder absolut erfolgen. Der Parameter *MoveSpeed* bestimmt die Geschwindigkeit der Verfahrbewegung.

- `public int Stop(int DeviceNumber){ ... }`
  Bricht die Abarbeitung eines an die NPM-Maschine gesendeten Befehls ab.

- `public double[] GetPosition(int DeviceNumber){ ... }`
  Liefert in einem Feld aus Double-Werten die absolute Position des Messtisches in den Koordinaten X, Y und Z.
  ```

Der Quellcode der Klasse NPMRemoteClass wurde als DLL kompiliert. NPMRemoteClass kann in diesem Zustand aber beispielsweise nur von einem Windowsprogramm auf demselben Rechner instanziiert werden. Um sie auch anderen Prozessbereichen zugänglich zu machen, bietet das .NET Framework die Möglichkeit die Klasse remotefähig zu machen. Dazu wird NPMRemoteClass von der *MarshalByRefObject* abgeleitet:

```
public class NPMRemoteClass : MarshalByRefObject
{ ... }
```

Dies allein ist aber noch nicht ausreichend. Ein Serverprozess muss für dieses Objekt noch einen *Uniform Resource Identifier* (URI) bereitstellen. Dafür kann beispielsweise der IIS verwendet werden. Die Verwendung des IIS als Aktivierungsagent hat allerdings zur Folge, dass für den Kommunikationskanal nur das http-Protokoll eingesetzt werden kann (Kapitel 7.1.3.2). Um aber den Vorteil von .NET Remoting, die Verwendung des schlankeren tcp-Protokolls, ausnutzen zu können, wird diese Variante nicht verwendet. Dementsprechend wurde ein Programm geschrieben, welches die Klasse serverseitig aktiviert:

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;

class NPMServerClass
{
      static void Main(string[] args)
      {
            TcpServerChannel channel = new TcpServerChannel(60622);
            ChannelServices.RegisterChannel(channel);

            RemotingConfiguration.RegisterWellKnownServiceType(
                                    typeof(NPM_DLL.NPMRemoteClass),
                                    "NPM",
                                    WellKnownObjectMode.SingleCall);

            Console.WriteLine("Press Enter to terminate...");
            Console.ReadLine();
      }
}
```

Zunächst legt das Serverprogramm einen Kanal an. Dieser stellt eine Leitung zwischen dem remotefähigen Objekt und den Clients her. Der angegebene Port ist frei aus dem Bereich der dynamischen Ports (dynamic ports: 49152 – 65535) wählbar und in einer Firewall evtl.

freizugeben. Anschließend wird der Kanal registriert. Der dritte Schritt registriert die Klasse NPMRemoteClass unter dem URI *NPM*. Der dritte Parameter für die Registrierung der Klasse gibt an, ob für jeden Aufruf der Klasse eine neue Instanz angelegt werden soll (*SingleCall*) oder ob nur eine Instanz der Klasse für alle Aufrufe angelegt wird (*Singleton*). Solange das Serverprogramm läuft, können Instanzen der Klasse auf entfernten Clients erzeugt und ihre Methoden aufgerufen werden.

Der Quellcode eines Clientprogramms muss, um nun auf die remotefähige Klasse zugreifen zu können, folgende Zeilen enthalten:

```
TcpClientChannel channel = new TcpClientChannel();
ChannelServices.RegisterChannel(channel);
RemotingConfiguration.RegisterWellKnownClientType(
                                        typeof(NPM_DLL.NPMRemoteClass),
                                        "tcp://141.24.37.32:60622/NPM");
NPM_DLL.NPMRemoteClass NPM = new NPM_DLL.NPMRemoteClass();
```

Zunächst wird wiederum ein Kanal angelegt und registriert. Diesmal vom Typ *TcpClientChannel*. Anschließend wird die remotefähige Klasse registriert und ein Objekt angelegt. Wichtig ist aber, dass innerhalb der Projektmappe des Visual Studio auf eine Kopie der kompilierten Klasse verwiesen wird (NPM_DLL). Diese muss zuvor auf den Client kopiert werden. Ohne diese Vorlage kann kein Proxy auf dem Client angelegt werden. In Bild 39 wird deshalb zusammengefasst, welche Dateien und Programme notwendig sind, um mittels .NET Remoting eine Klasse zur Steuerung der NPM-Maschine von einem entfernten Client aus anzusprechen.

Das im vorangegangenen Abschnitt beschriebene Windowsprogramm *Handsteuerung* stellt einen Client dar, der auf die Klasse NPMRemoteClass mittels .NET Remoting zugreift. Mit diesem Programm ist demonstriert worden, wie mit dieser Technologie, die Hardware der Nanopositionier- und Nanomessmaschine über das Internet hinweg gesteuert werden konnte. Zum Vergleich mit der Realisierung als XML-Webdienst wurde als Basis wiederum das Windowsprogramm Handsteuerung verwendet. Im Unterschied zur eben beschriebenen Realisierung wird die Kommunikation zwischen Client und Server durch einen Webdienst sichergestellt. Die Beschreibung dieser Realisierung erfolgt im nächsten Kapitel.

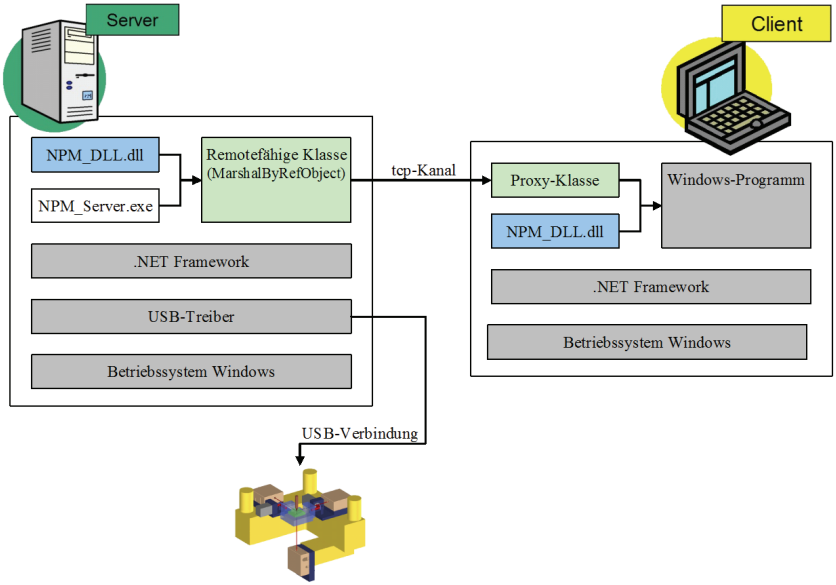


Bild 39: Struktur und notwendige Dateien der Handsteuerung mit .NET Remoting

7.2.2 Realisierung als XML-Webdienst

Analog zu der Variante einer Handsteuerung für NPM-Maschinen über das Internet mit .NET Remoting, wird bei der Realisierung mit XML-Webdiensten zunächst von der lokalen Klasse zur Steuerung der Maschine ausgegangen. In diesem Fall wurde die Klasse *Webservice_NPM* genannt:

```
public class Webservice_NPM
{ ... }
```

Die zu veröffentlichenden Methoden der Klasse heißen bei XML-Webservices *Webmethoden*. Prinzipiell gibt es in diesem Beispiel dieselben Methoden wie in der Variante mit .NET Remoting. Allerdings müssen die Methoden zusätzlich durch das Attribut *WebMethod* gekennzeichnet sein. Die Implementierung in C# sieht am Beispiel der Funktion *CountDevices* wie folgt aus:

```
[WebMethod(Description="Anzahl vorhandener Geraete ermitteln")]
public int CountDevices(){ ... }
```

Anhand des Attributes *WebMethod* kann der Compiler erkennen, welche Methoden veröffentlich werden sollen. Weiterhin können dem Attribut weitere Parameter übergeben werden. Im Beispiel ist das der Parameter *Description*, der eine kurze Beschreibung der Funktion liefert.

Auch die gesamte Klasse wird durch ein Attribut als Webdienst deklariert. Zusätzlich muss die Klasse von der Klasse *WebService* abgeleitet werden. In C# sieht die Klassendefinition wie folgt aus:

```
[WebService(  Description="Webdienst zur Handsteuerung der NPM-Maschine",
              Namespace="TUI/NPM",
              Name="Handsteuerung_NPM")]
public class Webservice_NPM : System.Web.Services.WebService
{ ... }
```

Die Parameter *Namespace* und *Name* sind notwendig, um den Webdienst später eindeutig lokalisieren zu können.

Die endgültige Bereitstellung der Methoden für entfernte Clients wird durch den IIS realisiert. Bei Verwendung des Visual Studio 2005 ist dieser Vorgang komfortabel automatisiert und für den Programmierer unproblematisch. Um die Vorgänge ein wenig zu verdeutlichen, muss an dieser Stelle nochmals kurz auf das Visual Studio eingegangen werden. Für die Erstellung eines Webdienstes wird dem Programmierer ein entsprechender Projekttyp angeboten. Nach Auswahl dieses Projekttyps, wird innerhalb des Verzeichnisbaumes des IIS, in der Regel ist dies das Verzeichnis C:\Inetpub\wwwroot, ein neues Verzeichnis erstellt. Der Name des Verzeichnisses entspricht dem Namen des Projektes. Standardmäßig wird der Name dieses Verzeichnisses auch Bestandteil des URL für die Identifikation des Webdienstes. Aus Sicht des IIS wird ein *virtuelles Verzeichnis* mit dem Namen des Projektes angelegt. In diesem Ordner werden verschiedene Dateien erzeugt. Da das Visual Studio konsequent die Code-Behind-Programmierung anwendet, entstehen dabei sowohl ASP.NET-Dateien, als auch Dateien, die den Quellcode enthalten. Die wichtigsten Dateien zum Verständnis von Webdiensten sind:

- *<WebServiceClassName>.asmx*
  Diese Datei bildet den Startpunkt des Webdienstes. Wird der URL dieser Datei aufgerufen, stellt der IIS die Webmethoden zur Verfügung. Der Name dieser Datei ist mit dem Namen der Webdienst-Klasse identisch.

- *<WebServiceClassName>.asmx.cs*
  Diese Datei enthält den Quellcode der Anwendung.

- *Global.asax* und *Global.asax.cs*
  Eine Datei und die dazugehörige Quellcodedatei zur Behandlung von globalen Ereignissen.

- *Web.config*

  Eine xml-basierte Datei für die Konfiguration des Webdienstes, z. B. die Steuerung von Authentifizierungsmechanismen.

Wird das Webservice-Projekt kompiliert, wird innerhalb des eben beschriebenen Ordners ein weiteres Unterverzeichnis mit dem Namen *bin* erstellt. In diesem Verzeichnis wird der kompilierte Zwischencode in Form einer DLL mit Namen <*ProjectName*>.*dll* abgelegt.

Über die Eingabe des URL der asmx-Datei in einen Webbrowser, kann der Webdienst gestartet werden. Im Internet Explorer sieht die Darstellung der Webmethoden für einen Webdienst namens *Webservice_NPM.asmx* und einem virtuellen Verzeichnis namens *Handsteuerung_NPM* wie folgt aus:

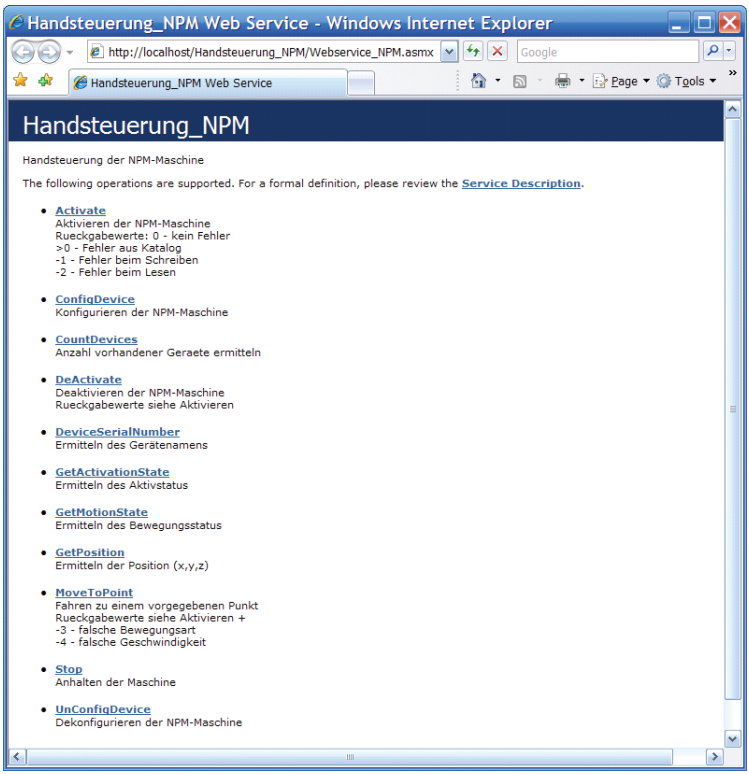


Bild 40: Startseite des Webdienstes im Internet Explorer

Damit stehen die Methoden im Internet anderen Clients zur Verfügung. Ein Client, welcher die Methoden implementieren möchte, muss nun entsprechende Anfragen an den URL senden. Der Aufruf der Funktion *CountDevices* sieht beispielsweise wie folgt aus:

```
http://localhost/Handsteuerung_NPM/Webservice_NPM.asmx?op=CountDevices
```

Die Antworten des Webdienstes sind, wie bereits im Kapitel 7.1.3.3 erklärt, Nachrichten in Form von XML-Daten. Der Client muss die Antworten entsprechend auswerten und weiterverarbeiten. Bei einer angeschlossenen NPM-Maschine sieht die XML-Antwort des Servers beispielsweise so aus:

```
<?xml version="1.0" encoding="utf-8" ?>
  <int xmlns="TUI/NPM">1</int>
```

Um einen Client in Form eines Windows-Programms mithilfe des Visual Studio 2005 zu erstellen, muss dem Projekt eine *Web-Referenz* auf den Webdienst hinzugefügt werden. Das Visual Studio bietet ein entsprechendes Hilfsmittel in Form einer Art *Webdienst-Browser* an. Gibt man hier den URL des Webdienstes und einen Namen für die Referenz ein, wird der Webdienst als Referenz in das Projekt eingebunden. Bild 41 zeigt den Webdienst-Browser:

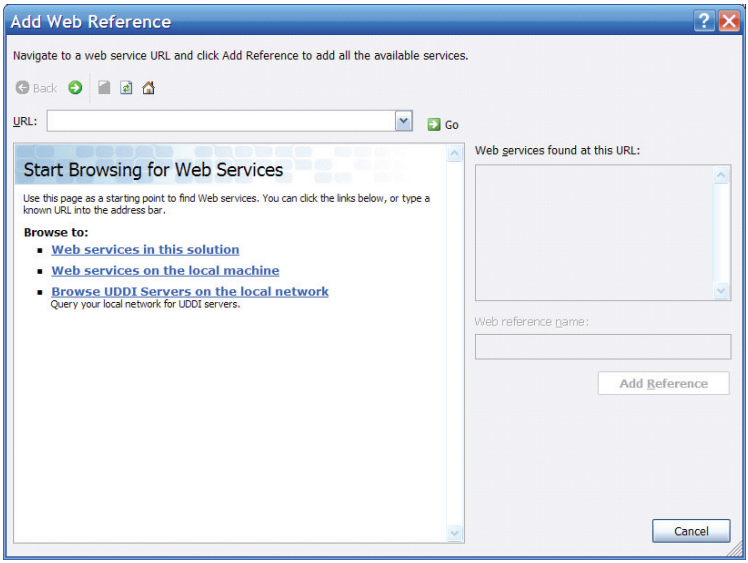


Bild 41: Der in das Visual Studio integrierte „Webdienst-Browser“

Diese Information wird in den Projekteigenschaften gespeichert. Damit ist der Zugriff auf die WSDL-Schnittstelle des Webdienstes sichergestellt und das Visual Studio kann

Informationen über die Klasse und deren Methoden ermitteln. Der Zugriff auf die Methoden erfolgt durch Instanziierung eines Objektes der hinzugefügten Referenz:

```
WebRef_NPM.Handsteuerung_NPM NPM = new WebRef_NPM.Handsteuerung_NPM();
int i = NPM.CountDevices();
```

Das Visual Studio ist ein ideales Werkzeug, um die Technologie XML-Webdienste zu nutzen. Dabei werden beide Richtungen, die Erstellung und die Nutzung eines Webdienstes, gleichermaßen berücksichtigt.

7.2.3 Vergleich beider Lösungen

Wie zuvor beschrieben, ließ sich eine Handsteuerung für eine NPM-Maschine sowohl mit der Technologie .NET Remoting als auch mittels XML-Webdienste realisieren. Aufgrund der identischen Bedienerschnittstelle erkennt der Anwender keine Unterschiede während der Benutzung. Aus programmiertechnischer Sicht sind jedoch folgende Unterschiede wichtig:

Plattformunabhängigkeit

Eine vollständige Plattformunabhängigkeit lässt sich nur mit XML-Webdiensten realisieren. Webdienste nutzen XML als Datenformat für den Informationsaustausch. Dieses Datenformat ist auf allen Hardware- und Softwareplattformen implementiert und kann entsprechend genutzt werden. Ein Client, der einen Webdienst referenziert, kann also in jeglicher Rechnerumgebung realisiert werden. Im Gegensatz dazu ist das Vorhandensein des .NET Frameworks beim Einsatz von .NET Remoting sowohl server- als auch clientseitig unumgänglich. Zwar hat sich in letzter Zeit eine Projektgruppe unter dem Namen MONO [56] mit der Realisierung eines Gegenstückes zur Microsofts .NET befasst, eine vollständige Kompatibilität kann aber nicht sichergestellt werden, da die vollständigen, internen Vorgänge des .NET Frameworks nicht zugänglich sind.

Transparenz

Die Transparenz in der Benutzung der Methoden zur Steuerung der NPM-Maschine ist bei beiden Technologien ähnlich groß. Ist erst mal ein Objekt der Klasse auf dem Client angelegt, erfolgen die Methodenaufrufe analog zur Programmierung auf einem lokalen Rechner. Aus persönlicher Sicht ist die Implementierung eines Webdienstes mithilfe des Visual Studio einfacher als die Registrierung eines tcp-Kanals und die Registrierung einer Proxyklasse bei .NET Remoting. Die Registrierung erfordert viel mehr Kenntnisse über den Server als dies bei Webdiensten der Fall ist. Bei XML-Webdiensten genügt das Wissen über den URL des Dienstes. Überdies muss die kompilierte DLL der Klasse bei .NET Remoting zunächst auf

den Clientrechner kopiert werden. Dies ist für die Wartbarkeit des Systems überaus nachteilig. Bei Änderungen im Quellcode einer Funktion muss die DLL dann wiederum allen Clients zur Verfügung gestellt werden.

Geschwindigkeit

Aufgrund der geringen Datenmenge, die für die Realisierung der Handsteuerung übertragen werden musste, ließen sich in diesem Anwendungsfall keine Geschwindigkeitsunterschiede zwischen beiden Technologien erkennen. Aufgrund der jeweiligen Methode zur Datenübertragung ist aber zu vermuten, dass das schlankere Protokoll des tcp-Kanals von .NET Remoting bei großen Datenmengen Geschwindigkeitsvorteile gegenüber Webdiensten bringt. Deshalb wurde eine Untersuchung durchgeführt, um den Einfluss der verwendeten Technologie auf die Übertragungsgeschwindigkeit bei großen Datenmengen zu messen.

Dazu wurde eine einfache Funktion programmiert, die ein Feld von Double-Werten erwartet, die einzelnen Feldelemente quadriert und das manipulierte Feld als Antwort zurücksendet.

```
public double[] Calculate(double[] ValuesToCalculate) { ... }
```

Anschließend wurde die Funktion als .NET Remoting Marshal-By-Reference-Object und als Webmethode veröffentlicht. Durch zwei entsprechende Clients wurde die Funktion nach folgendem Schema mehrfach aufgerufen:

- Die Anzahl der Feldelemente wurde beginnend bei einem Element bis hin zu 4100 Elementen des Typs Double variiert. Das entspricht einer Feldgröße von 8 Byte bis 32800 Byte.

- Je Feldgröße wurde der Methodenaufruf 50-mal wiederholt, um einen hinreichend genauen Mittelwert bilden zu können.

Da der Methodenaufruf in beiden Fällen asynchron erfolgte, setzt sich die *Antwortzeit* aus den drei Anteilen Senden der Anfrage, Berechnung der Feldelemente und Übertragung der Antwort zusammen. Eine zuvor durchgeführte Messung der Zeit, die für die Quadrierung der Feldelemente benötigt wird, hat allerdings gezeigt, dass dieser Zeitanteil vernachlässigt werden kann. Demzufolge wurde die Zeit, die zwischen der Anforderung und dem Eintreffen der Antwort verstrichen war, als *Antwortzeit* betrachtet. Die so definierte Antwortzeit ist also die doppelte Zeit, die für die Übertragung einer definierten Datenmenge benötigt wird. Für die so ermittelten Daten wurden die Durchschnittswerte berechnet und in einer Exceltabelle gespeichert. Die Ergebnisse der Untersuchung sind in Bild 42 zu sehen.

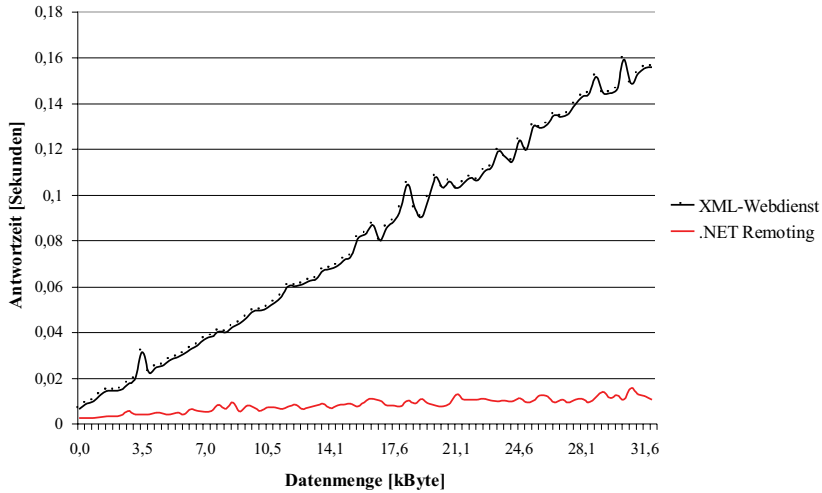


Bild 42: Untersuchung zur Datenübertragungsrate bei .NET Remoting und XML-Webdiensten

Zwischen beiden Technologien ist ein deutlicher Unterschied zu erkennen. Bei Datenmengen von etwa 32 kByte beträgt der Unterschied in etwa Faktor Zehn. Da aber die Datenpakete von der NPM-Maschine maximal 32 kByte groß sein können, muss dieser Unterschied durchaus als signifikant betrachtet und somit berücksichtigt werden. Der Grund für diesen deutlichen Unterschied liegt im verwendeten Übertragungsprotokoll. .NET Remoting verwendet ein schlankes Protokoll auf Basis von TCP/IP. Webdienste verpacken die Daten im XML-Format, was für einen deutlichen Zuwachs der zu übertragenden Datenmenge führt. In Bezug auf die Geschwindigkeit lässt sich also eindeutig ein Vorteil bei .NET Remoting erkennen.

Prinzipiell sind beide Technologien dazu geeignet, verteilte Anwendungen auf Basis des .NET Framework zu erstellen. Je nach Blinkwinkel treten Vor- und Nachteile in den Vorder- oder Hintergrund. Wie die Literatur aber zeigt, wächst die Bedeutung von Webdiensten immer weiter an, sodass sich eine klare Vorrangstellung von Webdiensten gegenüber .NET Remoting erkennen lässt. Dazu haben der Technologie nicht zuletzt auch die wichtigen Vorteile in der Konfiguration (Problem der Portnummern, Firewall-Konfiguration) und die Verwendung des xml-basierten SOAP-Protokolls zur Datenübertragung verholfen.

8 Weiterführende Aufgaben

Im Rahmen dieser Dissertation wurden verschiedene Komponenten einer modularen Steuerung für Nanopositionier- und Nanomessmaschinen entworfen und als Prototypen realisiert:

- Ein Programmeditor erlaubt auf grafischem Weg die Erstellung von Messprogrammen für die NPM-Maschine.
- Die Abarbeitung dieser Programme ist durch einen ersten Interpreter getestet und demonstriert worden.
- Für den Nachweis der Realisierbarkeit der Teleservicefunktionalität wurden ein Diagnosemodul und eine Handsteuerung entwickelt.

Für die Realisierung der genannten Komponenten wurde eine Reihe weiterer Programme, beispielsweise das Programm zur Ermittlung der Datenübertragungsgeschwindigkeit (Seite 99) oder das Programm zur Visualisierung der Tischposition im Zusammenhang mit der Handsteuerung aus Kapitel 7.2. entwickelt.

Die bestehenden Programme liefern einen wichtigen Beitrag zur Entwicklung einer modularen Steuerung für NPM-Maschinen. Dennoch bleibt Raum für weitere Aufgaben in diesem Themengebiet, die im Folgenden vorgeschlagen werden:

Messprogrammeditor und Interpreter

Der Messprogrammeditor ist in der jetzigen Realisierungsphase in der Lage auf der Basis des in XML vorliegenden Maschinenbefehlssatzes Messprogramme für NPM-Maschinen zu erstellen. Diese, ebenfalls im XML-Format gespeichert, können durch einen Interpreter abgearbeitet werden.

Eine geplante Überarbeitung des Editors sieht vor, den Befehlssatz mithilfe einer *Object-Factory* als Liste von Objekten des *Typs Instruction* abzubilden. Damit wäre eine Serialisierung in das XML-Format leichter realisierbar. Das aufwendige Suchen und Editieren in den DOM-Objekten von Befehlssatz und Messprogramm könnte so evtl. umgangen

werden. Auch eine Überarbeitung der Bedienoberfläche durch den Einsatz andockbarer Fenster ist Gegenstand zukünftiger Arbeiten.

Der Interpreter sollte durch die Erweiterung des XSD-Schemas der Befehlssatzdatei flexibler gestaltet werden. Auch die Integration weiterer Sicherheitsmechanismen im Fehlerfall ist notwendig.

Visualisierung

Ein weiterer wichtiger Teil innerhalb einer modularen Maschinensteuerung ist die Visualisierung verschiedenster Informationen. Zum einen besteht die Notwendigkeit stets über die Zustände einzelner Gerätekomponenten informiert zu sein. Als Beispiele seien hier die aktuelle Position des Messtisches oder die Lage von Messsystem und -objekt zueinander zu nennen. Andererseits ist es sinnvoll die Ergebnisse eines Messvorganges *online* verfolgen zu können. Ein entsprechendes Modul sollte daher direkt die Ergebnisse des Interpreters in anschaulicher Weise darstellen. Das bereits dafür erstellte Programmmodul stellt derzeit nur die Position des Messtisches im Raum dar. Weitere Arbeiten an diesem Programmteil müssen es dem Benutzer ermöglichen, auch die Geometrien von Messobjekt und Messsystem in die Darstellung einblenden zu können, um so abschätzen zu können, ob eine Kollision droht.

Ebenfalls in den Bereich der Visualisierung fällt die Online-Überwachung der Maschine. Es müssen weitere Programmteile entstehen, die eine Simulation von Messprogrammen erlauben oder die die aktuell gemessenen Messdaten sukzessiv, in Echtzeit und dreidimensional darstellen.

Und schließlich muss eine modulare Steuerung auch über den Zustand der Maschine Aufschluss geben: Welche Temperatur haben die Antriebe? Ist die Kommunikation im Netzwerk gestört? Treten sonstige Fehler auf? Die dafür notwendigen Daten werden zwar durch das Diagnosemodul bereits ermittelt, die Fülle der Informationen ist aber für die Überwachung eines Messprogramms zu umfangreich. Die notwendigen Zustandsdaten müssen also durch geeignete Module, nach Wichtigkeit und Anwendungsfall gefiltert, dargestellt werden können.

Archivierung

Bei der täglichen Nutzung einer NPM-Maschine sammelt sich schnell eine Vielzahl unterschiedlichster Dateien an, deren Bedeutung und Inhalt schnell verloren gehen, wenn nicht durch geeignete Hilfsmittel eine Möglichkeit geschaffen wird, zurückliegende Arbeiten systematisch zu archivieren. So sind es beispielsweise oft nur kleine Veränderungen an den

Koordinaten, in denen sich zwei Messprogramme unterscheiden. Daher wäre es sinnvoll, bereits erfolgreich durchgeführte Messprogramme recherchieren zu können, sie entsprechend abzuändern, als neues Programm zu speichern und abzuarbeiten. Dazu ist es notwendig, ein Werkzeug zu besitzen, um gespeicherte Messprogramme nach bestimmten Merkmalen zu durchsuchen. Dies ist ein Ziel weiterer Arbeiten im Zusammenhang mit der Archivierung von Messprogrammen.

Ebenfalls von zentraler Bedeutung ist das Auffinden bestimmter Messergebnisse. Diese könnten in einer Datenbank nach bestimmten Kriterien geordnet werden und somit schnell zur Verfügung gestellt werden. Wichtigstes Merkmal dabei wird die Visualisierung der Messwerte anhand eines Vorschaubildes sein. Aber auch andere Merkmalsdefinitionen, wie beispielsweise Messergebnisse mit bestimmten Oberflächenmerkmalen, etwa charakteristische Werkstückkanten und Höhenunterschiede, können zur Sortierung und Recherche von Messergebnissen hilfreich sein.

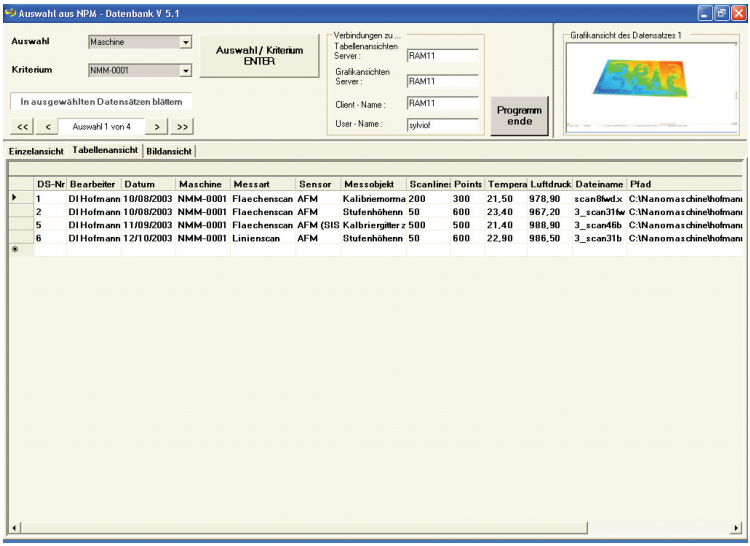


Bild 43: Oberfläche eines Programms zur Archivierung von Messwertdateien

Es ist daher auch Teil weiterführender Aufgaben, zu untersuchen, wie die Struktur der entstehenden Messprogramme und Messwertdateien gewählt und gegebenenfalls geändert werden muss, um spezifische Informationen über die jeweiligen Daten zu hinterlegen. Im Rahmen des SFB 622 ist ein Demoprogramm entstanden, welches genau diese Daten auswertet. Bild 43 zeigt die Bedienoberfläche des Demoprogramms. Es wurde als Windows-

Anwendung in der Programmiersprache C# geschrieben. Im Hintergrund arbeitete eine SQL-Datenbank. Mithilfe eines Update-Werkzeuges können die Daten aus den Messwertdateien in die Datenbank eingetragen werden. Durch die Auswahl verschiedener Kriterien werden nur die Dateien ermittelt, die diesen Kriterien entsprechen. Ein Vorschaubild erleichtert die visuelle Wiedererkennung von Messwertdateien.

Dieses Programm muss kontinuierlich an die Veränderungen im Dateiaufbau angepasst und durch zusätzliche Funktionen erweitert werden. Dabei kann die Aufgabe beliebig komplex ausgeweitet werden. So ist es beispielsweise denkbar, durch verschiedene Programme Oberflächenmerkmale wie Kanten, Stufen oder andere Formen aus den Messwerten zu extrahieren und somit Klassen von Messobjekten zu bilden.

9 Zusammenfassung

Die Nanopositionier- und Nanomessmaschine stellt ein einzigartiges Werkzeug für die Positionierung, Vermessung und, mithilfe geeigneter Nano-Werkzeuge, Bearbeitung von Objekten im metrologischen Bereich des Nanometers dar. Die dabei erreichbare hohe Auflösung, die geringe Messunsicherheit und der große Arbeitsbereich ermöglichen eine Vielzahl an unterschiedlichsten Anwendungsfällen.

Um aber den gesamten Funktionsumfang der NPM-Maschine ausschöpfen zu können, müssen geeignete Programme zur Steuerung entwickelt werden. Die bisher existierenden Softwarekomponenten bestanden aus vielen Einzelkomponenten. Ein durchgängiges Arbeiten an einer einheitlichen Programmoberfläche war somit nicht möglich. Weiterhin war die Erweiterbarkeit der Softwarelösungen nur schwer möglich. Die vorliegende Dissertation zeigt die Ergebnisse der Entwicklung von Programmkomponenten im Zusammenhang mit dem Entwurf einer modularen Steuerung für NPM-Maschinen. Dabei sollten die Modularität und eine damit verbundene spätere Erweiterbarkeit im Vordergrund stehen.

Der Befehlsumfang der NPM-Maschine ist in der zentralen Echtzeitsteuerung, der DSP-Einheit, hinterlegt. Zurzeit existieren 58 Befehle für unterschiedlichste Aufgaben und mit den verschiedensten Parametern. Um für die Steuerung der NPM-Maschine ein einheitliches Datenaustauschformat festzulegen, wurde der Maschinenbefehlssatz erstmals in Form einer XML-Datei gespeichert. Damit ist es möglich, den Befehlssatz jederzeit zu erweitern oder mithilfe verschiedener XML-Techniken an zukünftige Erfordernisse anzupassen. Das Datenformat XML bietet zudem aufgrund seiner großen Verbreitung eine große Zukunftssicherheit.

Auf der Basis des XML-Maschinenbefehlssatzes wurde ein grafischer Programmeditor zur Erstellung von Messprogrammen für NPM-Maschinen entwickelt. Als Messprogramme werden dabei sowohl die Programme verstanden, die der Vermessung eines Objektes mit der NPM-Maschine dienen, als auch solche, die beispielsweise lediglich für die Konfiguration der Maschine eingesetzt werden. Die mit dem Messprogrammeditor erstellten Programme werden ebenfalls im XML-Format gespeichert. Ein entsprechender Interpreter arbeitet die

Messprogramme ab und übersetzt sie in für die Maschine verständliche Bytefolgen. Dabei werden vom Messprogramm-Interpreter auch der Programmablauf und das Auftreten von Fehlern überwacht. Im Falle entstehender Messwerte, beispielsweise bei der Vermessung einer Oberfläche, werden die Messwerte wiederum in XML-Dateien gespeichert.

Ein weiterer wichtiger Punkt der modularen Steuerung von NPM-Maschinen ist die Realisierung von Teleservicefunktionen. Sowohl aus messtechnischen als auch aus organisatorischen Gründen ist eine Fernsteuerung verschiedener Funktionen der Steuerung notwendig. Daher wurde untersucht, welche Technologien für die Realisierung einer verteilten Anwendung in Frage kommen können. Als zwei der derzeit wichtigen Technologien haben sich .NET Remoting und XML-Webdienste herausgestellt. Für beide Techniken wurde der Funktionsnachweis anhand von Beispielapplikationen erbracht. Mithilfe eines Webdienstes wurde zudem ein leistungsfähiges Diagnosemodul zur Erfassung und Manipulation eingestellter Maschinenparameter entwickelt. Für zukünftige Arbeiten haben sich Webdienste als die Technologie herauskristallisiert, die für die Erweiterung der Teleservicefunktionalitäten von NPM-Maschinen eingesetzt werden soll.

In Rahmen der Arbeiten im Sonderforschungsbereich 622 wurden im Fachgebiet Rechneranwendung im Maschinenbau weitere Programme entwickelt, die durch die Ergebnisse der Dissertation beeinflusst wurden. Beispielsweise existiert ein Visualisierungswerkzeug zur dreidimensionalen Darstellung von Messergebnissen. Dieses basiert auf der Auswertung von XML-Dateien. Ebenfalls in diesem Zusammenhang ist eine erste Datenbankanwendung zu sehen. Sie dient der Archivierung von Messwerten durchgeführter Messungen und Messprogrammen. Anhand geeigneter Kriterien ist dadurch das Auffinden zurückliegender Messungen oder von Messprogrammen, die als Vorlage für neue Messaufgaben dienen sollen, erleichtert worden.

Für zukünftige Arbeiten an der modularen Steuerung von NPM-Maschinen ist die Integration weiterer Programmmodule vorgesehen. So ist beispielsweise eine Online-Darstellung von Messabläufen vorgesehen. Die Ergebnisse der vorliegenden Dissertation leisten somit bereits einen Beitrag zur Verbesserung der Bedienbarkeit von NPM-Maschinen und zeigen weitere Aufgaben für die Erweiterung einer modularen Steuerung auf.

10 Verzeichnisse

10.1 Literaturverzeichnis

[1] Advanced Micro Devices Inc. (AMD) (Hg.): 90 nm, 65 nm and 45 nm Process Nodes. 2007.
Online verfügbar: http://fab36.amd.com/en-US/tech_silicon_nodes.aspx

[2] Baum, B.: Die CGI-Schnittstelle. Verbindung des WWW mit existierender Software, Techniken und Werkzeuge, maschinelle Erzeugung von HTML, Verknüpfung des WWW mit Datenbanken. Fachhochschule Wedel. 1996.
Online verfügbar: http://www.fh-wedel.de/~si/seminare/ws96/ausarbeitung/cgi/cgi0.htm

[3] Bengel, G.: Grundkurs Verteilte Systeme. Grundlagen und Praxis des Client-Server-Computing; inklusive aktueller Technologien wie Web-Services u.a.; für Studenten und Praktiker. (Aus dem IT erfolgreich lernen). Vieweg, Wiesbaden. 2004.

[4] Bishop, J. M.; Horspool, N.: C# concisely. Pearson Education Ltd. – Addison-Wesley, Harlow. 2004.

[5] Braunschweig, M.; Frank, Sebastian; Weiß, Mathias: Teleservice für NPM-Maschinen. Tagungsunterlagen: 50. Internationales Wissenschaftliches Kolloquium, 19. bis 23.09.2005. Ilmenau. 2005.

[6] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E.; Yergeau, F.: Extensible Markup Language (XML) 1.0 (Third Edition). 2004. World Wide Web Consortium.
Online verfügbar: http://www.w3.org/TR/2004/REC-xml-20040204/REC-xml-20040204.xml

[7] CAMTECH GmbH (Hg.): EdgeCAM.
Online verfügbar: http://www.camtech.de/produkte/edgecam/edgecam.htm

[8] Chappell, D.: .NET verstehen. Einführung und Analyse – Der kompetente Überblick über die Möglichkeiten von .NET. (Programmer's choice). Addison-Wesley, München. 2002.

[9] Clark, J.; DeRose, S.: XML Path Languge (XPath). Version 1.0. 1999. Herausgegeben von World Wide Web Consortium.
Online verfügbar: http://www.w3.org/TR/xpath

[10] Czwalina, C.: Richtlinien für Zitate, Quellenangaben, Anmerkungen, Literaturverzeichnisse u.ä. Czwalina, Hamburg. 1997.

[11] Delcam (Hg.): PowerMILL.
Online verfügbar: ftp://arrow.delcam.com/pub/pdf/brochures/powerMILL.pdf

[12] Deutsches Institut für Normung (Hg.): Allgemeine Grundbegriffe. Berlin. Beuth. (Deutsche Norm, Teil 1319). 1985.

[13] Frank, S.; Braunschweig, Marion; Weiß, Mathias: Teleservice für Nanopositionier- und Nanomessmaschinen. Maschinenbau und Nanotechnik – Hochtechnologien des 21. Jahrhunderts: 47. Internationales Wissenschaftliches Kolloquium, 23. bis 26.09.2002. Ilmenau. 2002.

[14] Frank, S.; Braunschweig, Marion; Weiß, Mathias: Bedienung von NPM-Maschinen – Programmmodule und Schnittstellen. Tagungsunterlagen: 50. Internationales Wissenschaftliches Kolloquium, 19. bis 23.09.2005. Ilmenau. 2005.

[15] Frischalowski, D.: Visual C 2005. (Master class). Addison-Wesley, München. 2006.

[16] Gates, B.: Remarks by Bill Gates, Forum 2000. Redmond, Washington, 2000.

[17] Goldfarb, C. F.; Prescod, P.: Das XML-Handbuch. Addison-Wesley, München. 2000.

[18] Grieser, C.; Karras, L.: CALYPSO auf Fremdsystemen. In: Innovation SPEZIAL Messtechnik. 2007. Nr. 9, S. 59.
Online verfügbar:
http://www.zeiss.de/C1256CEE005B6986/0/E3912306D4985902C12572D500391859/$file/innovation_ms9d_s59_calypso-fuer-alle.pdf

[19] Grünbeck, T.: Telediagnose für Nanopositionier- und Nanomessmaschinen (NPM) als Webservice. Diplomarbeit. Technische Universität Ilmenau. 2004.

[20] Gunnerson, E.: C#. Die neue Sprache für Microsofts .NET-Plattform. (Galileo computing). Galileo Press, Bonn. 2001.

[21] Hausotte, T.: Nanopositionier- und Nanomessmaschine. Verl. ISLE, Ilmenau. 2002.

[22] ISO 8879: 1986: Information processing – Text and office systems – Standard Generalized Markup Language (SGML). 1986.

[23] Jäger, G.; Manske, E.; Hausotte, T.; Büchner, H.-J.; Grünwald, R.: A novel approach to positioning- and measuring technique for nanometrology. 2002.
Online verfügbar: http://www.db-thueringen.de/servlets/DocumentServlet?id=7271

[24] Jäger, G.; Manske, E.; Hausotte, T.; Füßl, R.; Grünwald, R.; Büchner, H.-J.: Nanomess- und -positioniertechnik. In: Maschinenbau und Nanotechnik – Hochtechnologien des 21. Jahrhunderts. 2002, S. 566–567

[25] Jäger, G.; Manske, E.; Hausotte, T.: Neue Anwendungen der Nanomessmaschine (NPM-Maschine) durch die Entwicklung nanoskaliger optischer und taktiler Tastsensoren. In: Technisches Messen. 2006. Nr. 9, S. 457–464.
Online verfügbar: http://www.atypon-link.com/OLD/doi/pdf/10.1524/teme.2006.73.9.457

[26] Jamal, R.; Hagestedt, A.: LabVIEW für Studenten. Pearson Studium, München. 2004.

[27] Jones, A.: Visual C#.NET – Programmier-Rezepte. Hunderte von Lösungen und Codebeispielen aus der Praxis. Microsoft Press, Unterschleißheim. 2004.

[28] Kuhrmann, M.; Calamé, J.; Horn, E.: Verteilte Systeme mit .NET Remoting. Grundlagen – Konzepte – Praxis. Elsevier/Spektrum Akad. Verl., Heidelberg. 2004.

[29] Kurzgefasst: Was ist Nanotechnik? In: Die Zeit. Jg. 2004. Ausgabe 23, 27.5.2004, S. 44

[30] Kuschke, M.; Wölfel, L.: Web Services kompakt. Spektrum Akad. Verl., Heidelberg. 2002.

[31] Langner, T.: Web-basierte Anwendungsentwicklung. Die wichtigsten Technologien für Webapplikationen im Überblick. Elsevier Spektrum Akad. Verl., München. 2004.

[32] Mathematisch Technische Software-Entwicklung GmbH (Hg.): Topcam Topturn Topmill. CNC Komplettbearbeitung Drehen Fräsen.
Online verfügbar: http://www.mts-cnc.com/deutsch/pdf/Produktion_de.pdf

[33] MATLAB. Die Sprache für technische Berechnungen.
Online verfügbar: http://www.mathworks.de/products/matlab/

[34] Metasprache verbindet Wissenschaften. In: Freies Wort. Jg. 2002. 12.06.2002, S. 12

[35] Microsoft: Namespaces and Selected Classes in the .NET Framework.
Online verfügbar: http://msdn.microsoft.com/vstudio/previous/2003/posters/posterfiles/Namespaces_Selected_Classes_X0910395pst_a_OL.pdf

[36] Nanotechnologie. nano-polis, Institut für Innovationsgestaltung im Nanozeitalter. 2004.
Online verfügbar: http://www.nano-polis.de/nanotechnologie.htm

[37] Neumann, J.: Lockere Kuppeleien. Interfaces sinnvoll einsetzen. In: dotnetpro. 2005. Jg. 2005, Nr. 6, S. 16–19

[38] Open Mind Technologies AG (Hg.): hyperMILL. Die zukunftsweisende CAD/CAM-Lösung.
Online verfügbar: ftp://ftp.openmind-tech.com/OMWEB/pdf_dt/hMILL_CAD_2007_D.pdf

[39] Prosise, J.: Microsoft .NET - Entwicklerhandbuch. (Entwicklerbuch). Microsoft Press, Unterschleißheim. 2002.

[40] Schäpers, A.; Huttary, R.; Bremes, D.: C# - Kompendium. Windows- und Web-Programmierung mit Visual Studio .NET. (Kompendium). Markt & Technik in Pearson Education Deutschland; Markt-und-Technik-Verl., München. 2004.

[41] Schwichtenberg, H.: Omnipräsent. Die Extensible Markup Language im .Net Framework 2.0. In: iX, Magazin für professionelle Informationstechnik. 2005. Nr. 8, S. 54–59

[42] Schwichtenberg, H.; Conrad, S.: Microsoft ASP.NET - das Entwicklerbuch. Microsoft Press, Unterschleißheim. 2002.

[43] Seltzer, L. J.: Remote Control Software. In: Windows 2000 Magazine. 2001. Nr. 4, S. 123–125.
Online verfügbar: http://www.windowsitpro.com/Files/20075/20075.pdf

[44] Siemens AG (Hg.): UGS NX.
Online verfügbar: http://www.ugsplm.de/produkte/nx/unigraphics/

[45] SIOS Messtechnik GmbH: Miniaturinterferometer mit Planspiegelreflektor. Serie SP. SIOS Messtechnik GmbH (Ilmenau).
Online verfügbar: http://www.sios.de/DEUTSCH/PRODUKTE/SP_D_FOT.PDF

[46] SIOS Messtechnik GmbH: Nanopositionier- und Nanomessmaschine. NMM-1. SIOS Messtechnik GmbH (Ilmenau).
Online verfügbar: http://www.sios.de/DEUTSCH/PRODUKTE/NMM.PDF

[47] Skulschus, M.; Wiederstein, M.: XSL-FO für PDF und Druck. Detaillierte Erläuterungen zu XSL-FO 1.0; Kombination mit XSLT und XPath ; Entwicklung wiederverwendbarer Komponenten. (Programmierung). mitp, Bonn. 2005.

[48] SL - Automatisierungstechnik GmbH (Hg.): SL-Programmiersystem EXSL. Produktinformation.
Online verfügbar: http://www.slhome.com/downloads/Produktinfo/Catalog/Deutsch/Produktinfo%20EXSL-WIN.pdf

[49] Srinivasan, R.: RPC: Remote Procedure Calls. Protocol Specification. Version 2. The Internet Engineering Task Force.
Online verfügbar: http://tools.ietf.org/html/rfc1831

[50] Tanenbaum, A. S.; van Steen, M.; Muhr, J.: Verteilte Systeme. Grundlagen und Paradigmen. (Pearson Studium Informatik, Verteilte Systeme). Pearson Studium, München. 2003.

[51] The Internet Engineering Task Force (Hg.): RPC: Remote Procedure Call. Protocol Specification. Version 2. Sun Microsystems, Inc.
Online verfügbar: http://tools.ietf.org/html/rfc1057

[52] van Venn, H. W. de: Internetbasierter Teleservice für mobile Maschinen und Anlagen. (Berichte aus der Automatisierungstechnik). Shaker, Aachen. 2003.

[53] Vasters, C. F.: Net-Crashkurs. C; NET Framework; ASP.NET; VB.NET; ADO.NET; Managed C++. Microsoft Press, Unterschleißheim. 2001.

[54] Warnecke, H.-J.; Dutschke, W.: Fertigungsmeßtechnik. Springer, Berlin u.a. 1984.

[55] Weiß, M.; Frank, Sebastian: Teleservice for nano-positioning and nano-measuring machines. Teleservis dlja nano-izmeritel'nych i nano-pozicionnych masin. Varna (Bulgarien). 2004.

[56] What is Mono?
Online verfügbar: http://www.mono-project.com/Main_Page

[57] World Wide Web Consortium (Hg.): Web Service Description Language (WSDL). Version 2.0. Part 0: Primer. World Wide Web Consortium.
Online verfügbar: http://www.w3.org/TR/2007/WD-wsdl20-primer-20070326/

[58] World Wide Web Consortium: W3C Publishes New Editions of Core XML Standards. Improvements clarify, complete foundation of XML family. 2006. World Wide Web Consortium.
Online verfügbar: http://www.w3.org/2006/07/xml-pressrelease.html.en

[59] World Wide Web Consortium: SOAP Version 1.2. Part 1: Messaging Framework. World Wide Web Consortium.
Online verfügbar: http://www.w3.org/TR/soap12-part1/

[60] Wyke, R. A.; Rehman, S.; Leupen, B.: XML - Das Entwicklerbuch. Microsoft Press, Unterschleißheim. 2002.

10.2 Abbildungsverzeichnis

10.3 Softwareverzeichnis

Für die Erstellung der Programmmodule und der Dissertation wurde die im folgenden aufgelistete Software eingesetzt. Als Betriebssystem diente Microsoft Windows XP SP2.

Office-Programme

- Microsoft Office Word 2000, Microsoft Office Word 2003
- Microsoft Office Excel 2000, Microsoft Office Excel 2003
- Microsoft Office PowerPoint 2000, Microsoft Office PowerPoint 2003
- Microsoft Office Visio 2003

Programmierumgebungen

- Microsoft Visual Studio .NET, Visual Studio .NET 2003
- Microsoft Visual Studio 2005
- Microsoft .NET Framework 1.0, .NET Framework 1.1, .NET Framework 2.0

Alle Programme wurden in der Programmiersprache C# erstellt.

Grafikbibliothek

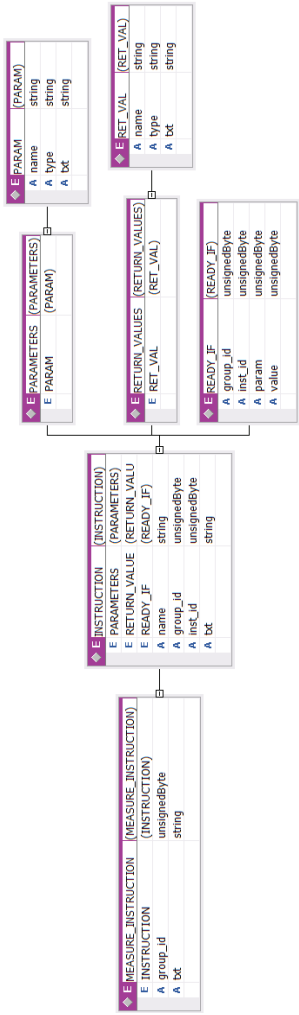
- CsGL, Version 1.4.1, http://csgl.sourceforge.net/index.html

Bildbearbeitungsprogramme

- Microsoft Photo-Editor
- Microsoft Paint
- Paint .NET, http://www.getpaint.net/index2.html
- Microsoft Visual Studio

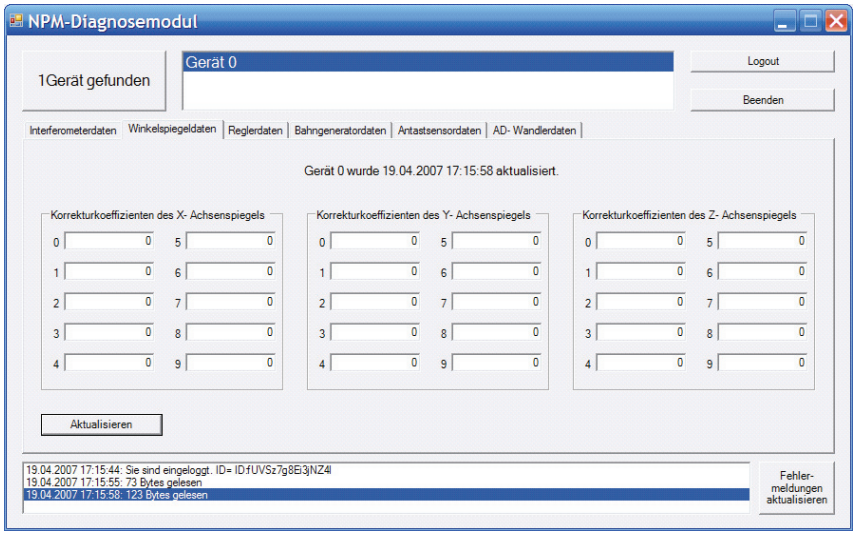
Anhang Programmfenster, Diagramme

Kapitel 5.4

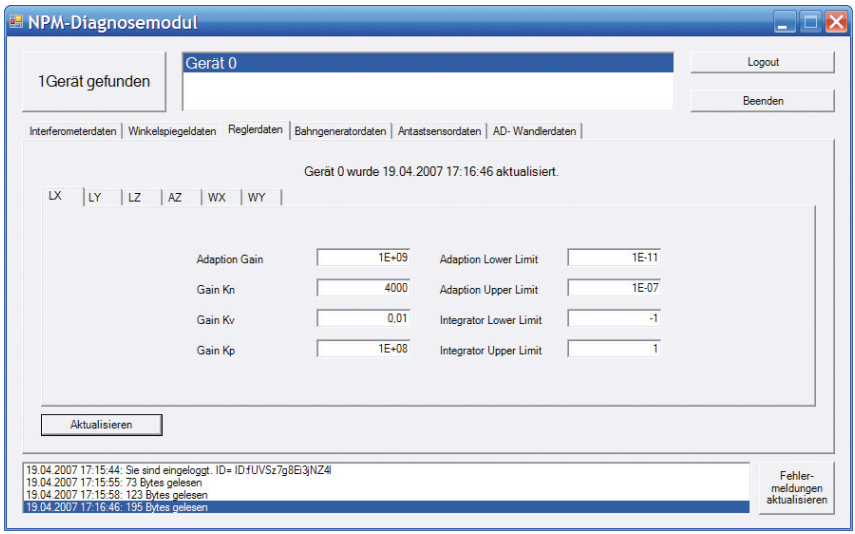


XSD-Schema der Elemente vom Typ MEASURE_INSTRUCTION

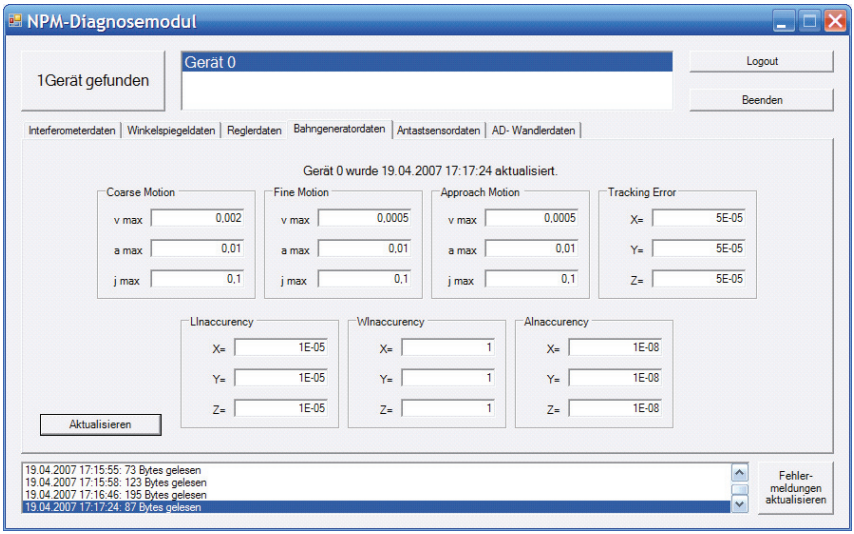
Kapitel 6.3.2



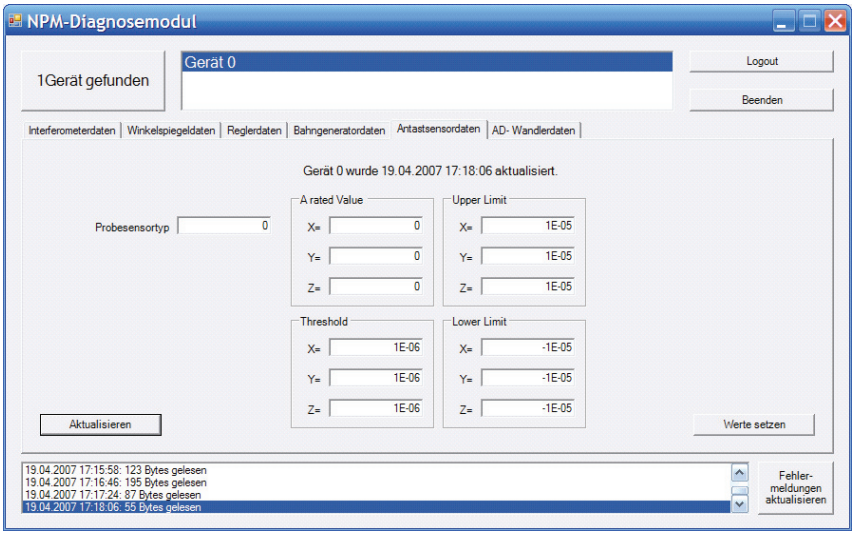
Bedienoberfläche des Diagnosemoduls, Registerkarte Winkelspiegeldaten (siehe Kapitel 6.3.2)



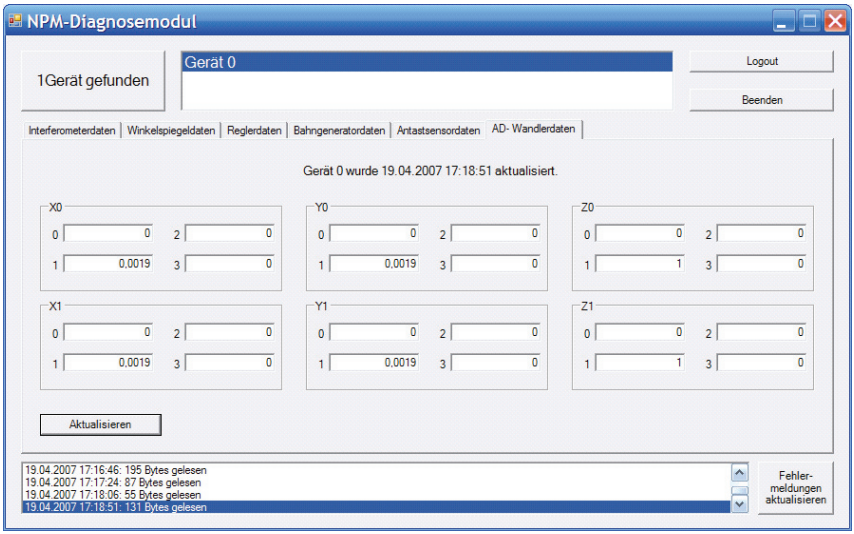
Bedienoberfläche des Diagnosemoduls, Registerkarte Reglerdaten (siehe Kapitel 6.3.2)



Bedienoberfläche des Diagnosemoduls, Registerkarte Bahngeneratordaten (siehe Kapitel 6.3.2)

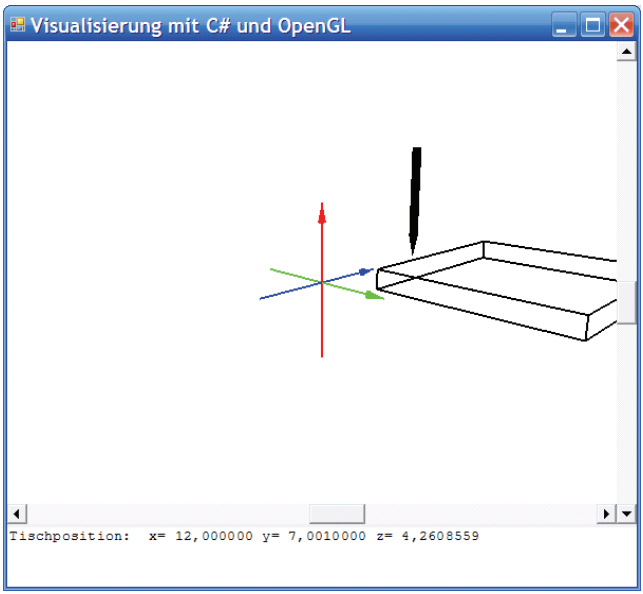


Bedienoberfläche des Diagnosemoduls, Registerkarte Antastsensordaten (siehe Kapitel 6.3.2)



Bedienoberfläche des Diagnosemoduls, Registerkarte AD-Wandlerdaten (siehe Kapitel 6.3.2)

Kapitel 7.2



3D-Echtzeitvisualisierung der Tischposition im Programm „Handsteuerung“ (Kapitel 7.2)