Private FernFachhochschule

Darmstadt

Fachbereich Informatik

# Analyse, Design und Realisierung eines webbasierten SCADA Systems mit OPC XML-DA Schnittstelle zu mehreren Soft-SPS Systemen

vorgelegt bei: Prof. Dr. Achim Gottscheber
von: Martin Barkhausen
Matr.-Nr.: 872021
Anschrift: Luisenstr. 85, 42103 Wuppertal
Abgabetermin: 1.9.2005

# Zusammenfassung

Um den Erfolg heutiger physikalischer Großexperimente zu sichern, ist es angesichts der Komplexität notwendig, für einen stabilen und sicheren Ablauf zu sorgen. Dies ist die Aufgabe der so genannten Slow Control Systeme. Sie überwachen permanent solche Parameter, die zu einer Beschädigung der Detektoren oder zu einer Beeinträchtigung oder Beeinflussung des Messbetriebes führen könnten.

Diese Systeme sollen auch durch das Bedienpersonal und anderen EDV-Systemen überwacht und gesteuert werden können. Zur Gewährleistung dieser Aufgabe wurden im Rahmen dieser Arbeit für die Fluoreszenzdetektoren im Pierre Auger Experiment ein so genannter *OPC XML-DA Gateway*, ein *OPC XML-DA Client* und ein *SCS-HTTP-Server* spezifiziert, entworfen und entwickelt.

Der OPC XML-DA Gateway bietet die Daten des Slow Control Systems, auf die über eine OPC DA Schnittstelle (DCOM) zugegriffen werden kann, über „SOAP over HTTP“ nach der Spezifikation OPC XML-DA der OPC Foundation an. Diese Schnittstelle bietet eine Interoperabilität und durch das zustandslose Protokoll auch eine Stabilität in fehlerhaften Netzwerken (wie z. B. unzuverlässige Richtfunkstrecken im oben genannten Experiment).

Der OPC XML-DA Client ist eine Komponente mit einer Programmier-schnittstelle für C++, die den vereinfachten Zugriff auf den OPC XML-DA Gateway und somit auch auf die Slow Control Daten ermöglicht.

Der SCS-HTTP-Server ist ein Visualisierungs- und Steuerungssystem (SCADA[1] System), der auf die Slow Control Systeme über die OPC XML-DA Schnittstelle zugreift. In gebräuchlichen Internetbrowsern können die durch den SCS-HTTP-Server erzeugten HTML-Seiten dargestellt werden.

Alle Komponenten sind parametrisierbar und nach Standardspezifikationen entworfen und entwickelt worden, so dass sie nicht nur in weiteren physikalischen Experimenten, sondern auch in industriellen Steuerungsanwendungen in heterogenen Systemen Relevanz haben.

[1] Technische Abkürzung für Supervisory Control and Data Acquisition.

# Abstract

In order to ensure the success of today's large physics experiments in view of necessarily complexity a stable and safe operational sequence is required. This is the task of the Slow Control Systems. Such systems permanently supervise parameters, which could lead to damage to the detectors or to the impairment or undue influence of the measuring enterprise.

These systems must be able to be supervised and steered by the service personnel and other electronic data processing system. In Order to guarantee the success of this task, in the context of this thesis, an OPC XML-DA gateway, an OPC XML-DA Client and an SCS-HTTP-server is specified, designed and developed for the fluorescence detectors in the Pierre Auger experiment.

The so called OPC XML-DA Gateway provides the data for the Slow Control System, which can be accessed over an OPC DA Interface (DCOM) using “SOAP over HTTP” according to the OPC XML-DA specification of the OPC Foundation. This Interface provides interoperability and, using a stateless protocol, stability in malfunctioning networks (e.g. unreliable radio links in the above experiment).

The so called OPC XML-DA Client is a component, with a programming interface for C++, which makes simplified access possible on the OPC XML-DA Gateway and thus to the Slow Control System’s data.

The so called SCS-HTTP-Server is a Supervisory Control and Data Acquisition (SCADA) System, which accesses the Slow Control System over the OPC XML-DA Interface. The HTML pages produced by the SCS-HTTP-Server can be represented in common Internet browsers.

All components were designed and developed to be parameterizable according to standard specifications, in order that they have relevance not only in further physical experiments but also in industrial control applications in heterogeneous systems.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# 1 Einleitung

## 1.1 Motivation

Nicht nur in der Physik werden heute Soft-SPS Systeme eingesetzt, sondern auch in industriellen Steuerungsanwendungen. Viele dieser Systeme bringen zwar schon eine Visualisierungs- und Steuerungssoftware mit, die meistens jedoch proprietär sind und oft nur auf einem Betriebssystem betrieben werden können.

Diesen Fall trifft man auch bei dem Soft-SPS System 4Control der Firma Softing an [SOFTI]. Bei diesem System gibt es eine Visualisierungs- und Steuerungssoftware. Diese Software kann aber nur mit dem Internet Explorer, einem Internet Browser der Firma Microsoft, betrieben werden. Und dieser steht nur für Windows-Betriebssysteme zur Verfügung. Außerdem kann mit der Visualisierungs- und Steuerungssoftware nur ein System abgebildet werden. In unseren Fall werden 4 Soft SPS Systeme betrieben, wodurch pro System ein Browser geöffnet und sichtbar auf dem Bildschirm abgelegt werden müsste.

Seit Juli 2003 gibt es den neuen Standard OPC XML-DA. Dieser Standard wird schon von einigen wenigen Soft-SPS Systemen angeboten [OPCXM04]. Ein Ansatz für das oben erläuterte Problem ist, diese Standard-Schnittstelle für eine parametrisierbare und konsolidierende Visualisierungs- und Steuerungssoftware zu benutzen.

## 1.2 Ziel der Arbeit

Das Ziel dieser Arbeit ist es, eine Visualisierungs- und Steuerungssoftware für 4 Soft-SPS Systeme in der Programmiersprache C++ zu entwickeln, wobei die Schwerpunkte auf Wiederverwendbarkeit und Interoperabilität der Komponenten gelegt werden sollen. Dabei soll auch eine Komponente entstehen, mit der andere Programme auf die Soft-SPS-Systeme zugreifen können.

Aus dem bestehenden Altsystem sollen in der Analysephase für das entstehende SCADA-System Anforderungen an die neuen Komponenten gestellt werden, wobei besonders auf die schon aufgetretenen Probleme - wie z. B. das unzuverlässige Netzwerk - geachtet werden soll. Aus diesen Anforderungen soll in der Designphase ein Konzept erstellt werden, wie die Komponenten zu benutzen und zu entwickeln sind. In der Implementierungsphase soll das Konzept umgesetzt und getestet werden.

## 1.3 Vorgehensweise

Zu Beginn der Arbeit gab es noch keine Vorstellungen, in welcher Technologie das Projekt realisiert werden sollte. Darum wurde sich bei der Entwicklung an das evolutionäre Vorgehensmodell (siehe Abbildung 1.1) gehalten.

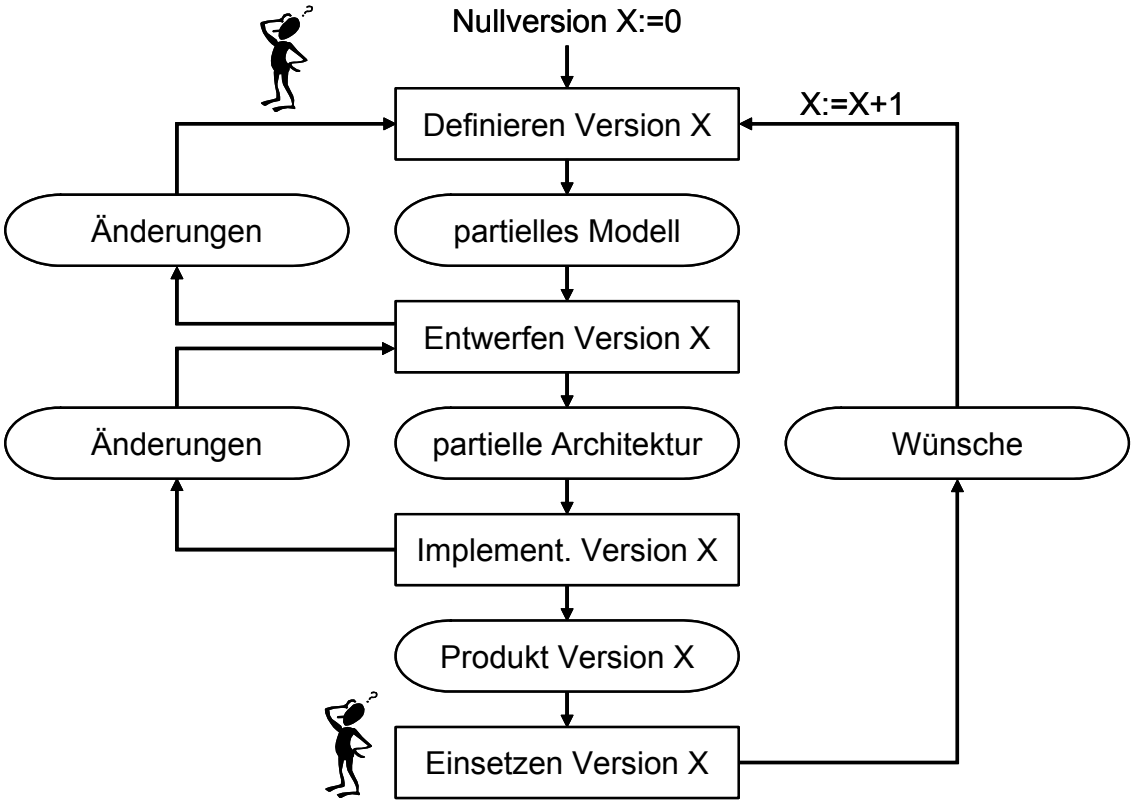


**Abbildung 1.1 Evolutionäres Vorgehensmodell [VORGE].**

Durch dieses Modell wird die Umsetzung in voller Breite vermieden, sondern es erfolgt zuerst eine Konzentration auf die Kernanforderungen und dann eine Annäherung an das immer mehr entstehende Ziel.

# 2 Astrophysikalische Grundlagen

## 2.1 Die kosmische Strahlung

Unter der kosmischen Strahlung versteht man geladene Teilchen (Protonen und Atomkerne), die in einem Energiebereich von ca. $10^7$ eV bis über $10^{20}$ eV auf die Erdatmosphäre treffen.

Zusätzlich zu den geladenen Teilchen treffen neutrale Teilchen - wie z. B. Photonen und Neutrinos - auf die Atmosphäre. Gamma- und Neutrino-Astronomie machen heutzutage ebenfalls einen beträchtlichen Teil der Forschung im Bereich der Astroteilchenphysik aus.

### 2.1.1 Geschichtliches

Zu Beginn des 20. Jahrhundert wurde zunächst eine von der Erde ausgehende ionisierende Strahlung vermutet, als man die Entladung von Elektroskopen beobachtete. Die Vermutung konnte auch mit Messungen auf dem Eiffelturm nicht richtig widerlegt werden. Im Jahre 1912 beobachtete Viktor Hess bei Ballonflügen bis in ca. 5000 m Höhe, dass die ionisierende Strahlung, nach einem leichten Rückgang bis ca. 1200 m, zu großen Höhen hin stark zunimmt. Er schloss daraus auf einen extraterrestrischen Ursprung der Strahlung und erhielt dafür 1936 den Nobelpreis [HESS12].

Kohlhörster und Bothe zeigten 1929 durch koinzidente Signale in übereinander liegenden Geiger-Müller-Zählrohren, dass es zumindest einen stark durchdringenden geladenen Anteil der Strahlung gibt.

Pierre Auger hat 1938 ebenfalls koinzidente Signale gemessen - dieses Mal allerdings mit Zählrohren, die in horizontaler Richtung bis über hundert Meter voneinander entfernt waren. Er folgerte daraus, dass diese Teilchen Sekundärprodukte aus Wechselwirkungen mit der Atmosphäre sind. Damit wurde Pierre Auger zum Entdecker der ausgedehnten Luftschauer [AUGER38].

Nach und nach konnten verschiedene (Elementar-)Teilchen - wie z. B. das Positron, das Myon oder das Pion in ausgedehnten Luftschauern nachgewiesen werden.

Durch die Verlagerung des Teilchennachweises an die großen Beschleuniger wie z. B. am CERN[2] verlor man in der darauf folgenden Zeit etwas das Interesse an der kosmischen Strahlung. Das änderte sich durch die Beobachtung von Luftschauern, die von Primärteilchen mit einer Energie von über $10^{20}$ eV ausgelöst wurden, die mit heutigen Beschleunigern nicht erreicht werden kann.

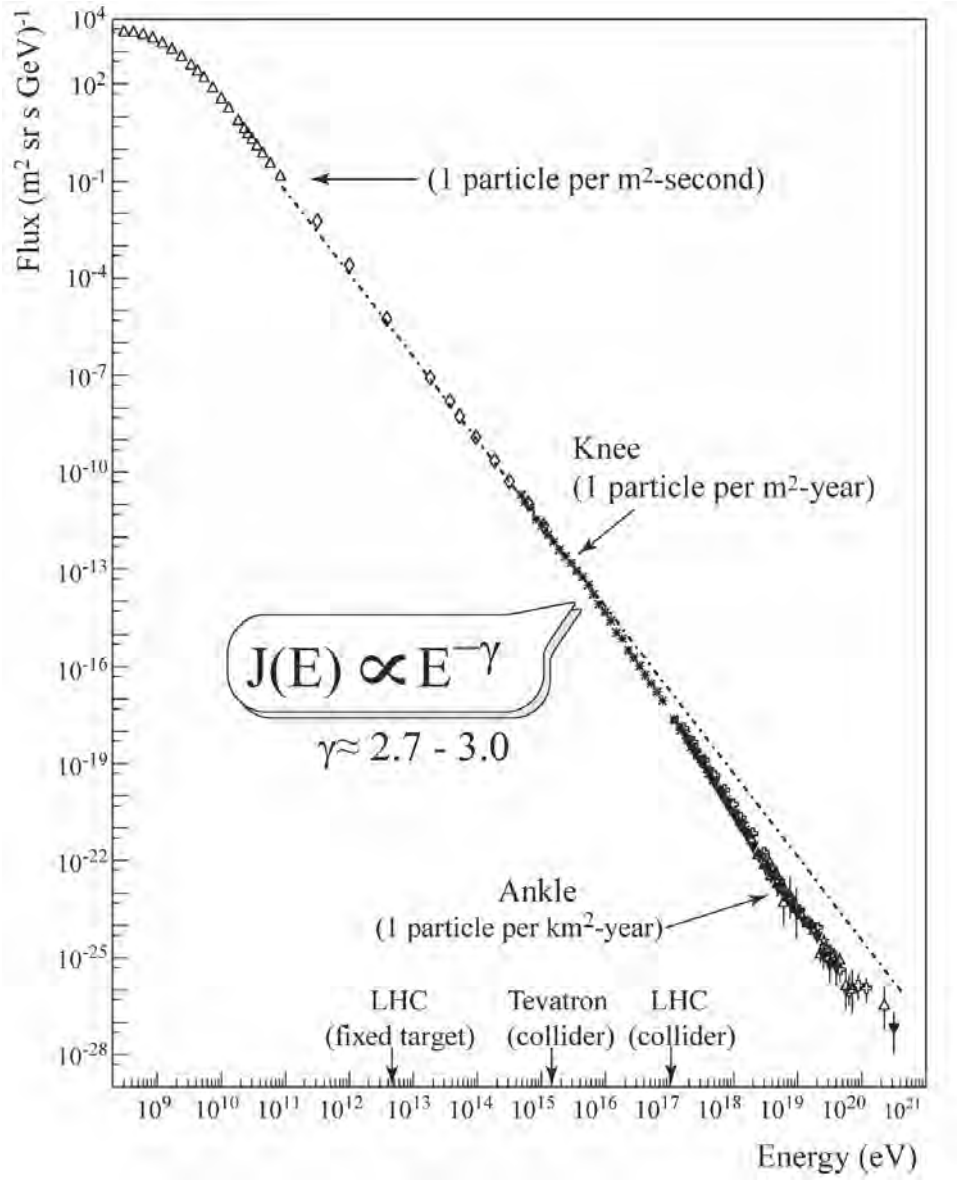


**Abbildung 2.1 Differenzielles Energiespektrum der kosmischen Strahlung [AUGER97].**

[2] Abkürzung für Conseil Européenne pour la Recherche Nucléaire (Europäische Organisation für Kernforschung).

### 2.1.2 Energiespektrum

Die Energie der kosmischen Strahlung deckt einen Bereich von fast 15 Größenordnungen ab. Die Energie eines Teilchens von $10^{20}$ eV entspricht der Energie eines Tennisballes mit einer Geschwindigkeit von ca. 100 km/h. Die Energie und der Fluss der Primärteilchen der kosmischen Strahlung hängen über ein Potenzgesetz (siehe Abbildung 2.1) zusammen.

Momentan gibt es eine Menge von Vermutungen, warum das Spektrum bei den höchsten Energien einen solchen Verlauf zeigt. Eine weitere Frage ist, was und wo die Quellen dieser Strahlung sind.

Eine Hauptaufgabe des Pierre Auger Experiments (siehe Kap. 2.3) ist die Klärung der Frage nach dem Ursprung und der Vermessung dieser Strahlung insbesondere bei den höchsten Energien.

## 2.2 Ausgedehnte Luftschauer

### 2.2.1 Phänomenologie

Ausgedehnte Luftschauer entstehen, wenn ein hochenergetisches Teilchen der kosmischen Strahlung mit Atomkernen der Erdatmosphäre in Wechselwirkung tritt. Dabei werden Sekundärteilchen produziert, welche ihrerseits wieder mit der Atmosphäre in Wechselwirkung treten und neue Teilchen erzeugen. So entsteht eine Kaskade, die bis zum Erreichen des Erdbodens aus mehreren Millionen Teilchen besteht. Da sich alle Teilchen mit annähernder Lichtgeschwindigkeit bewegen, besteht der Luftschauer aus einer etwa einen Meter dicken Scheibe (siehe Abbildung 2.2).

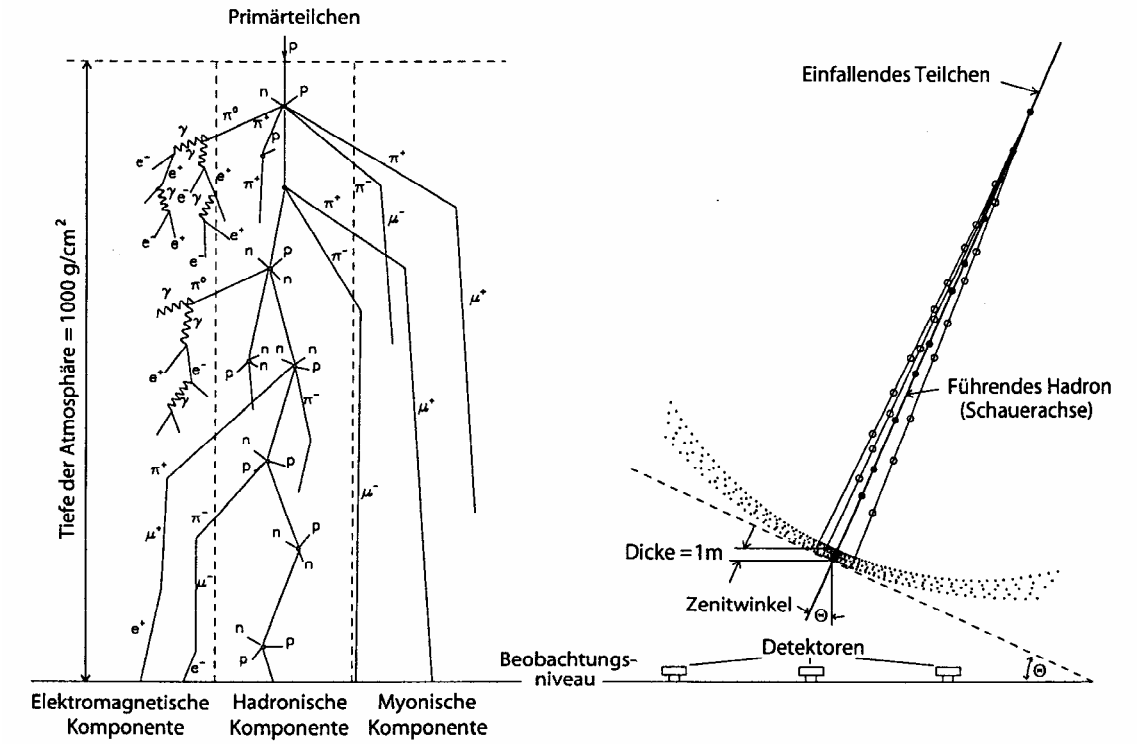


**Abbildung 2.2 Schematische Darstellung eines ausgedehnten Luftschauers [ALKHO75].**

## 2.2.2 Nachweis von Luftschauern

Wie in Kap. 2.1.2 bereits erwähnt, deckt die kosmische Strahlung einen Energiebereich von fast 15 Größenordnungen ab. Ab einer Energie von ca. $10^{13}$ eV werden in der Atmosphäre Luftschauer ausgelöst, deren Sekundärteilchen in signifikanter Anzahl den Erdboden erreichen. Für den Nachweis von Luftschauern kommen unterschiedliche Techniken zum Einsatz, die sich miteinander kombiniert auch sehr gut ergänzen können.

Der Nachweis der Sekundärteilchen erfolgt über großflächige Detektorarrays, die am Erdboden in einiger Entfernung voneinander aufgestellt werden. Mit den Sekundärteilchen wird dann der Schauer rekonstruiert. Dabei steigt die benötigte Fläche des Detektorarrays mit der Primärenergie, da einerseits die Ausdehnung der Luftschauer zunimmt und andererseits der Fluss zu höheren Energien hin stark abnimmt. Beispiele für solche Experimente sind KASCADE [DOLL90], AGASA [TAKED03] und das Pierre Auger Observatorium (siehe Kap. 2.3).

Die Sekundärteilchen eines Luftschauers lösen in der Atmosphäre auch Fluoreszenzleuchten durch Anregung von Stickstoffmolekülen aus. Dies ist erst ab einer Primärenergie von ca. $10^{17}$ eV nachweisbar. Unterhalb dieser Energie kann das Leuchten nicht vom Untergrund - wie z. B. Sternen- und Mondlicht - getrennt werden.

Das isotrop emittierte Fluoreszenzlicht wird seitlich zum Schauer mit sehr empfindlichen Teleskopen gemessen. Beispiele solcher Experimente sind das „Fly's Eye" [BIRD95] und das Pierre Auger Observatorium (siehe Kap. 2.3).

## 2.3 Das Pierre Auger Observatorium

Das Pierre Auger Observatorium ist ein zurzeit immer noch im Aufbau befindliches Experiment zur Messung von ausgedehnten Luftschauern. Es wurde für die Vermessung von höchstenergetischen Teilchen der kosmischen Strahlung mit Energien größer $10^{18}$ eV konstruiert. Es wird in internationaler Zusammenarbeit von 16 Nationen auf einer Fläche von ca. 3000 $km^2$ als das weltweit größte Experiment zur Untersuchung der kosmischen Strahlung errichtet. Standort ist die Pampa Amarilla in der argentinischen Provinz Mendoza, die sich durch besonders viel freie Fläche, wenig Streulicht von umgebenden Städten, unbelastete Atmosphäre und einen fast immer unbewölkten Himmel auszeichnet.

Das Observatorium besteht aus einem hybriden Detektorsystem, d.h. die Luftschauer werden gleichzeitig von zwei verschiedenen Detektortypen gemessen. Ein 3000 $km^2$ großes Detektorarray, mit 1600 Detektoren im Abstand von jeweils 1,5 km, gibt Auskunft über die Lateralverteilung der Teilchen im Luftschauer, während 24 Fluoreszenzteleskope in 4 FD[3]-Gebäuden am Rande des Detektorarrays die longitudinale Entwicklung des Schauers untersuchen (siehe Abbildung 2.3). Die Entfernung zwischen zwei benachbarten Detektorgebäuden beträgt ca. 45 km Luftlinie. Dieses Hybridsystem wurde gewählt, um unterschiedliche Ergebnisse aus zwei anderen Experimenten zu bestätigen bzw. zu widerlegen und um die Fehler zu minimieren und Daten besser rekonstruieren zu können.

[3] Abkürzung für Fluoreszenzdetektor.

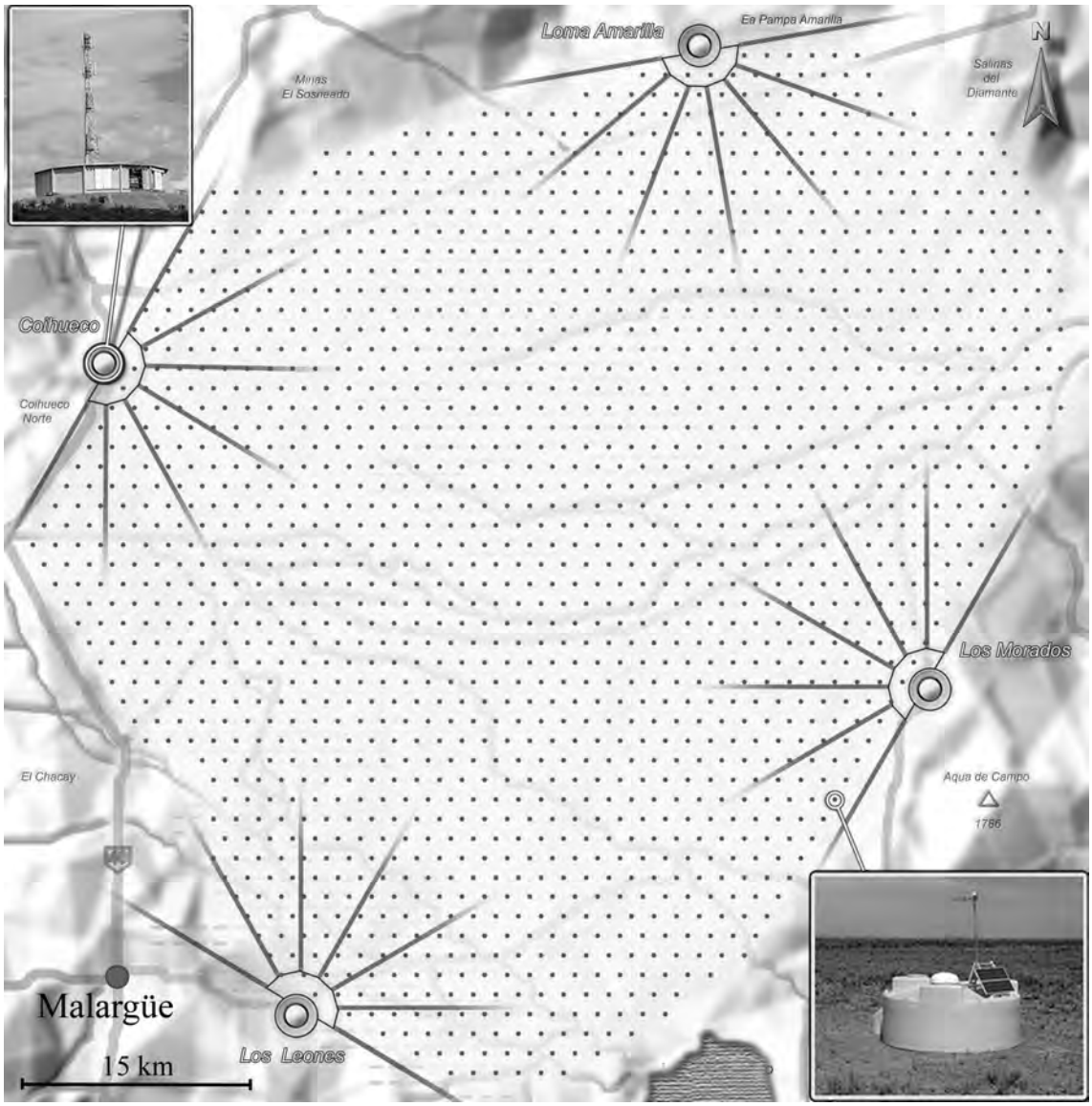


**Abbildung 2.3 Karte des Arrays, mit Fotos von einem FD-Gebäude (oben links) und einem SD-Detektor (unten rechts).**

Die erzeugten Daten werden über Richtfunkstrecken zum Kontrollraum, auch CDAS[4] genannt, in Malargüe geschickt.

In den vier Gebäuden am Rande des Detektorarrays befinden sich jeweils sechs der insgesamt 24 Teleskope. Ein Fluoreszenzteleskop besteht aus einem segmentierten sphärischen Spiegel und einer Pixelkamera, welche aus 440 Photomultipliern besteht. Diese können durch eine zu hohe Lichtintensität wie etwa das Sonnenlicht leicht zerstört werden.

[4] Abkürzung für Central Data Aquisition System (Kontrollzentrum des Pierre Auger Observatorium).

# 3 Technische Grundlagen

## 3.1 DCOM

DCOM steht für Distributed Component Object Model und beschreibt ein Objektmodell für das Implementieren verteilter Anwendungen entsprechend dem Client-Server-Paradigma. Es löste bzw. ergänzte die OLE (Object Linking und Embedding) Technologie der Firma Microsoft. Ein Client kann gleichzeitig mehrere Server nutzen, und ein Server kann seine Funktionalität mehreren Clients gleichzeitig zur Verfügung stellen.

Das „Herzstück“ von DCOM ist die Schnittstelle, über die die DCOM-Objekte ihre Dienste anbieten. Eine Schnittstelle beschreibt eine Gruppe von zusammengehörigen Funktionen. Mit der IDL (Interface Definition Language) wird die Syntax und Semantik der angebotenen Dienste festgelegt. Die Schnittstellen werden mit einem globalen, eindeutigen, 128 Bit langen GUID (Global Unique Identifier) signiert.

DCOM-Objekte existieren im DCOM-Server. Es gibt zwei prinzipielle Ausprägungen dieser Server: DLLs, die sog. InProc-Server, oder EXEs, die sog. OutProc-Server, die den eigentlichen ausführbaren Programmcode der Objekte beinhalten. Die DCOM-Objekte werden über den Class Identifier (CLSID) identifiziert. Der Client benutzt diese CLSID zum Anlegen des Objektes.

DCOM wurde von Microsoft für Windows-Betriebssysteme entwickelt. Es gibt aber auch Versuche von Portierungen auf andere Betriebsysteme, wie z. B. Linux.

## 3.2 HTTP - Hypertext Transfer Protocol

Das Hypertext Transfer Protocol (HTTP) ist ein zustandsloses Protokoll der Anwendungsschicht zur Übertragung von Hypermedia Informationen. Dabei wird auf Einfachheit und Geschwindigkeit geachtet [RFC2616].

HTTP basiert auf einem asymmetrischen Anfrage/Antwort-Paradigma. Dadurch ergibt sich für jede Verbindung die Rollenverteilung von Client und Server. Es sitzt auf dem darunter liegenden verbindungsorientierten TCP. Der Client schickt einen Request, und der Server antwortet mit einem Response. Die Nachrichtentypen bestehen aus einer Request-Zeile bzw. Status-Zeile, einem Header und dem Body, den eigentlichen Daten. In der Request-Line stehen die angefragte Methode, der URI [URI94] und die

Protokollversion. Die zwei wichtigsten und meist verwendeten Methoden sind GET und POST [RFC2616].

Die GET-Methode fordert die mit dem URI assoziierten Informationen oder, falls der URI einen Prozess angibt, die durch diesen produzierten Daten an. Dabei kann das Abfragen der Information von dem Datum der Quelle abhängig gemacht werden.

Mit POST werden Daten vom Client zum Server übertragen. Dies wird zur Aufnahme neuer Ressourcen, Nachrichten für Newsgroups usw. durch Übertragen von Formulardaten oder zum Einfügen in eine Datenbank benutzt. Die durch POST tatsächlich ausgelöste Aktion hängt in erster Linie vom Server ab und wird normalerweise von diesem anhand des URI entschieden. Das verschickte Objekt sollte dabei dem Request-URI untergeordnet sein. Der Client kann dabei ein URI für die neue Ressource vorschlagen. Es ist aber nicht notwendig, dass eine neue Ressource angelegt oder verfügbar gemacht wird. Die Antwort des Servers gibt darüber Auskunft.

## 3.3 Web Services

Allgemein ist ein Web Service ein Stück Software, das seinen Dienst über das Internet anbietet, mittels standardisierter XML[5] [XML] Schnittstellen angesprochen wird und weder an ein spezifisches Betriebssystem noch an eine spezifische Programmiersprache gebunden ist [Cer02].

Ein Web Service wird dabei als abstrakter Begriff angesehen, der durch einen konkreten Software-Agenten implementiert wird. Während der Service die zur Verfügung gestellte Funktionalität in abstrakter Form beschreibt, stellt der Agent die konkrete Entität dar, welche Nachrichten sendet und empfängt. So könnte ein Web Service z. B. zu einem Zeitpunkt durch einen Java Agenten und zum nächsten durch einen C++ Agenten implementiert sein, trotzdem bleibt der Web Service derselbe [BHM+04].

Für den Bereich des XML Messaging gibt es viele Alternativen, wie z. B. XML Remote Procedure Calls (XML-RPC), das SOAP (siehe Kap. 3.4) oder auch das Verschicken von XML Dokumenten mittels HTTP.

[5] Abkürzung für Extensible Markup Language.

## 3.4 SOAP

SOAP[6] ist ein Quasi-Standard für die XML Nachrichten, welcher derzeit vom W3C in der Version 1.2 verabschiedet wurde. Er ermöglicht Funktionsaufrufe über Rechner und Systemgrenzen hinweg [SOAP03].

Hierzu definiert SOAP, wie Funktionsaufrufe und Parameter in einer spezifischen XML Notation kodiert werden und wie die Anbindung an das Transportprotokoll (z. B. HTTP, SMTP [RFC821], FTP[RFC959]) erfolgt. Die meisten SOAP-Nachrichten werden mittels einer HTTP-POST-Methode an einen Webserver übertragen und zurückgeliefert. Man spricht in diesem Zusammenhang von „SOAP over HTTP“.

Betrachten wir nun den Aufbau einer SOAP-Nachricht am Beispiel von „SOAP over HTTP“ genauer. In Abbildung 3.1 rechts sehen wir eine solche Nachricht. Der obere Teil ist HTTP spezifisch. Der untere Teil enthält die eigentliche SOAP-Nachricht, die auch als SOAP-Payload bezeichnet wird. Für das HTTP-Protokoll ist der Payload nur ein sehr langer String.

Der SOAP-Payload enthält den Umschlag (envelope), welcher den Inhalt der Nachricht beschreibt. Dieser enthält das optionalen „Header“-Element, das z. B. für eine Authentifizierung genutzt werden kann, und das „Body“-Element mit den eigentlichen Daten. Diese Daten beinhalten entweder eine Anfrage (request) mit dem Namen der aufzurufenden Funktion auf dem Server des Service Providers, eine Antwort (response) mit dem Ergebnis eines Funktionsaufrufes oder eine Fehlerbeschreibung (fault) mit einer detaillierten Fehlerinformation.

Dazu enthält der Payload einen Satz von Vorschriften zur Umsetzung anwendungsspezifischer Datentypen und eine Vereinbarung, wie die entfernten Prozeduraufrufe und deren Antworten darzustellen sind.

Damit SOAP-Nachrichten bzw. Web Services einheitlich verwendet werden können, ist es notwendig, diese in einer konsistenten Art und Weise zu definieren. Dafür verwendet man die Web Service Definition Language (WSDL) [WSDL01].

Ein großer Nachteil von SOAP-Nachrichten gegenüber anderen Schnittstellen - wie z. B. RPC, DCOM (siehe Kap. 3.1) oder CORBA - ist der große Datenoverhead,

[6] SOAP war bis zur Version 1.1 eine Abkürzung für Simple Object Access Protocol, seit Version 1.2 ist SOAP jedoch offiziell keine Abkürzung mehr.

der durch die Verwendung von XML entsteht. Dies belastet nicht nur das Netzwerk, sondern verbraucht dadurch auch mehr Ressourcen beim Parsen und Verarbeiten.

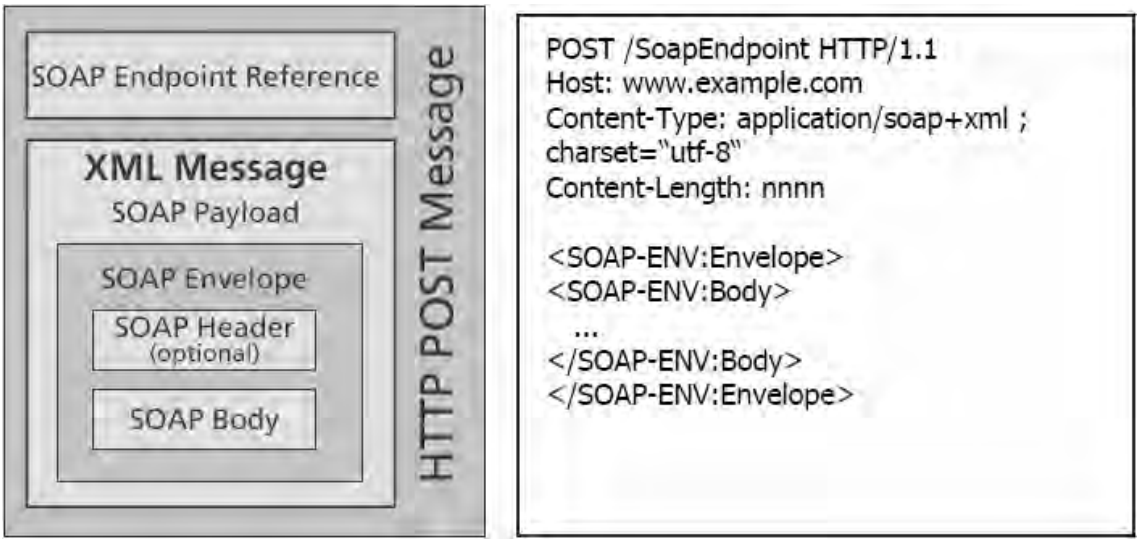


**Abbildung 3.1 Aufbau einer „SOAP over HTTP“-Nachricht.**

## 3.5 OPC

OPC war eine Abkürzung für “OLE for Process Control”. Heute steht OPC für „Openess, Productivity & Collaboration" und gibt weniger den Zusammenhang mit einer bestimmten Basistechnologie als vielmehr die Kennzeichen der offenen, interoperablen und produktiven OPC Schnittstelle wieder.

Unter diesem Begriff werden verschiedene Dienste für den Zugriff auf Informationen aus Prozessabläufen spezifiziert. Die Spezifikation unterscheidet dabei zwischen verschiedenen Grunddiensten, wie Variablenzugriff (OPC DA [OPCDA03]), Alarme (OPC AE [OPCAE02]), Datenarchivierung (OPC HDA [OPCHD03]) und Variablenzugriff über SOAP (OPC XML DA [OPCXM04]).

In diesem Dokument wird ausschließlich über die Spezifikationen OPC DA und OPC XML-DA informiert. Für diese Spezifikationen ist die OPC Foundation zuständig.

### 3.5.1 OPC Foundation

Eine wichtige Vorraussetzung für das Gelingen einer Standardisierungsinitiative ist eine die Interessen der beteiligten Mitglieder koordinierende Instanz. Dafür wurde 1996 die OPC Foundation als eine unabhängige non-profit Organisation gegründet, welche das

Ziel der Weiterentwicklung und Unterstützung des neuen Standards verfolgt [IWANI00].
Die wichtigste Aufgabe der OPC Foundation kommt der Erstellung von Spezifikationen zu.

### 3.5.2 OPC DA

Die historisch erste und meist benutzte Spezifikation ist die OPC DA Spezifikation. Der Hintergedanke für die Entwicklung der OPC DA Spezifikation war die Erstellung eines universellen Treibers für die Automatisierungstechnik. Dieser Treiber kapselt den eigentlichen Treiber und stellt die allgemein verständlichen Dienste über eine universelle Schnittstelle zur Verfügung.

Um eine möglichst große Unabhängigkeit zwischen der Prozessanbindung und der eigentlichen Anwendung zu erreichen, wird die Prozessanbindung durch eine eigenständige Anwendung realisiert, die als OPC DA Server bezeichnet wird. Eine Anwendung, die auf die Daten eines solchen Servers zugreifen will, wird als OPC DA Client bezeichnet.

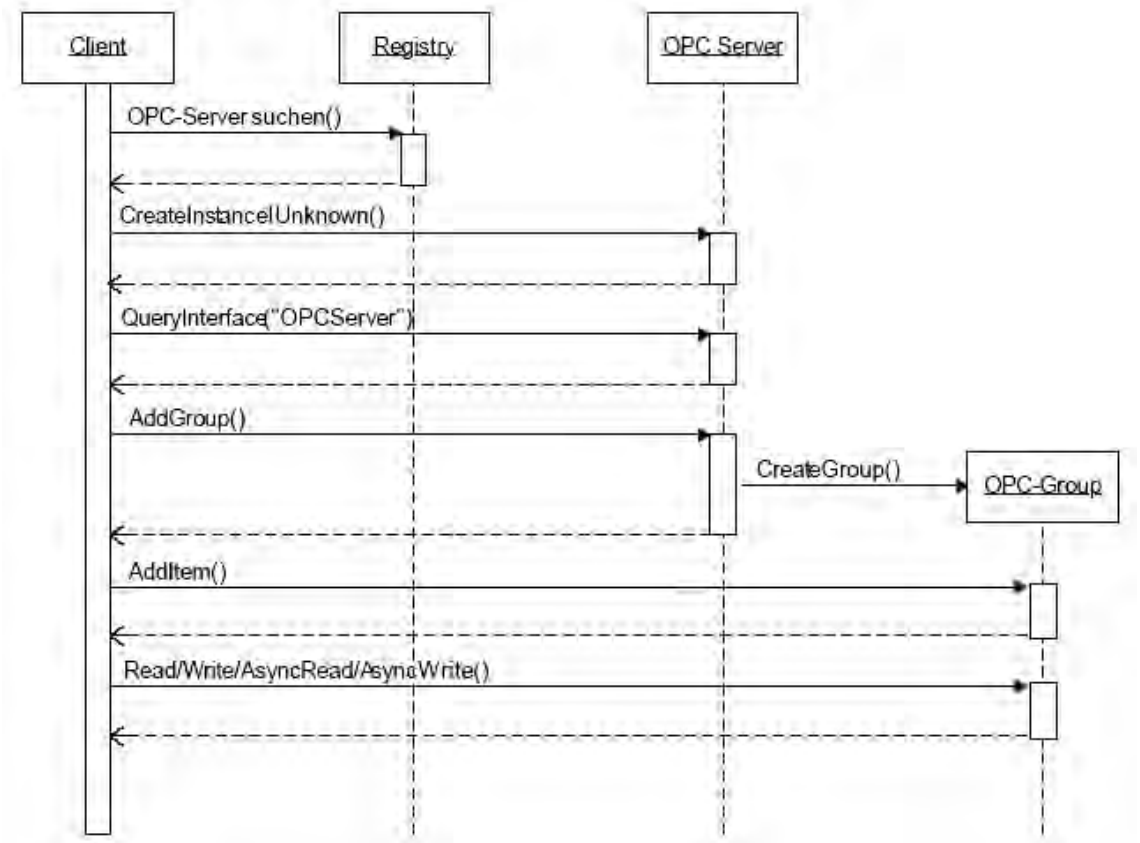


**Abbildung 3.2 Sequenzdiagramm für den Zugriff auf eine OPC DA Schnittstelle.**

Das Client-Server-Paradigma wurde gewählt damit die Server-Anwendung mehreren Clients ihre Dienste zur Verfügung stellen kann. Dafür wird die DCOM-Technologie (siehe Kap. 3.1) der Firma Microsoft eingesetzt.

Über das OPC DA Server Objekt wird der OPC DA Server verwaltet. Neben den Methoden zur Statusinformation und Fehlerbehandlung stellt das Objekt hauptsächlich Methoden zum Anlegen und Verwalten von Gruppen bereit, in denen wiederum Elemente angelegt werden können. Legt ein Client über das Server-Objekt eine neue Gruppe an, wird automatisch ein „Group"-Objekt für diese Gruppe erzeugt und an den Client übergeben. Über das „Group"-Objekt werden die Elemente in der Gruppe angelegt und verwaltet (siehe Abbildung 3.2). Außerdem existieren Methoden für Lese- und Schreibzugriffe auf die Elemente der Gruppe.

Ein Element repräsentiert die Prozessvariable, mit der es verbunden ist, innerhalb des OPC DA Servers.

Beim Anlegen einer Gruppe kann zwischen „aus der Hardware lesen" oder „aus dem Cache lesen" entschieden werden. Die erste Auswahl liefert immer den aktuellen Wert von der Hardware, die zweite ist dafür schneller.

Die Elementdatenbank bezeichnet man als Adressraum. Dieser ist hierarchisch aufgebaut. Für das Abfragen dieses Adressraums wird auch eine Schnittstelle bereitgestellt. Eine hierarchische Abfrage unterscheidet zwischen Knoten und Elementen, die mit Verzeichnissen und Dateien in einem Dateisystem verglichen werden können.

### 3.5.3 OPC XML-DA

OPC XML-DA wurde entwickelt um auch andere Betriebssysteme zu unterstützen und einen interoperativen Zugriff auf Daten über das Internet zu ermöglichen. Diese Spezifikation baut auf SOAP auf und kann somit auf jedem System verwendet werden, das HTTP und XML unterstützt.

Es gibt acht Methoden in der OPC XML-DA Spezifikation. Jede dieser Methoden unterteilt sich in einen Anfrage- (Request) und einen Anwortteil (Response). Der Request wird vom Client und der Response vom Server erzeugt.

Die Methoden „GetStatus" und „GetProperties" sind für das Abfragen des Serverstatus und der Elementeigenschaften zuständig. „Read" und „Write" sind für das Lesen und Schreiben der Daten zuständig. Die Methoden „Subscribe",

„SubscriptionPolledRefresh“ und „SubscriptionCancle“ bedienen die so genannte Subscription.

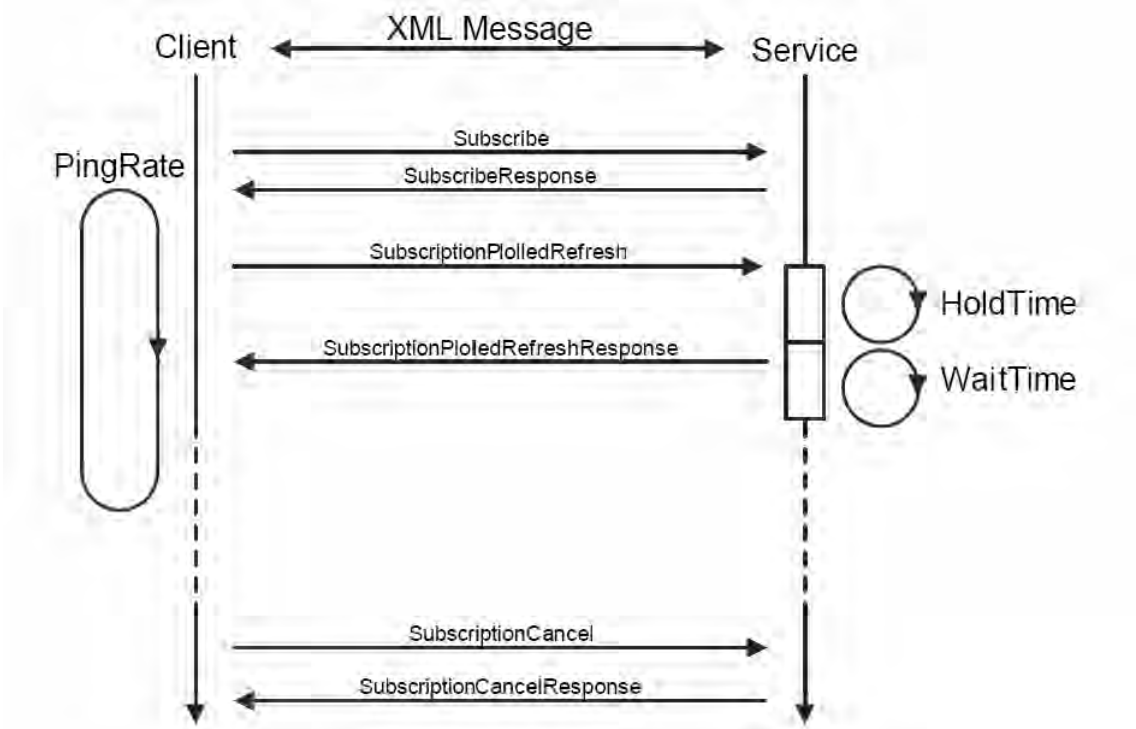


**Abbildung 3.3 Nachrichtenaustausch bei OPC XML-DA mittels der „Subcription“.**

Eine Subscription definiert eine Gruppe von Elementen, vergleichbar mit einem „Group“-Objekt der OPC DA Spezifikation (siehe Abbildung 3.3). Sie hat die drei wichtigen Parameter „PingRate“, „HoldTime“ und „WaitTime“. Die „PingRate“ ist die Zeit, die der Server zwischen zwei Clientanfragen wartet, bevor die Subscription automatisch vom Server aufgelöst wird. Bei einer „SubscriptionPolledRefresh“-Anfrage wartet der Server die Zeit, die in „HoldTime“ übergeben wurde. Hat sich in der Zwischenzeit ein Wert eines Elementes geändert, so wird vom Server direkt eine Antwort generiert; ansonsten wartet der Server mit dem Generieren einer Antwort, bis sich ein Wert ändert oder die „WaitTime“ abgelaufen ist. Mit der Hilfe von „HoldTime“ und „WaitTime“ können Änderungen registriert werden, ohne dass vom Client ein schnelles, aufeinander folgendes Abfragen initiiert wird.

Ein weiterer Parameter ist „DeadBand“. Mit diesem Parameter wird bei analogen Werten eine Prozentzahl festgelegt. Erst wenn die Änderung größer als der angegebene Prozentwert ist, wird dieser Wert als geändert registriert.

## 3.6 Das Slow Control System

Das Slow Control System sorgt für einen stabilen und sicheren Betrieb der Fluoreszenzteleskope. Sie übernehmen die permanente Überwachung von Parametern, die zu einer Beschädigung der Detektoren oder zu einer Beeinträchtigung oder Beeinflussung des Messbetriebes führen könnten. Im Bedarfsfall müssen vom Slow Control System entsprechende Maßnahmen - wie z. B. die automatische Abschaltung von Photomultipliern bei schädlich hoher Lichtintensität - ergriffen werden. Als weitere Aufgabe ist das System jederzeit über den aktuellen Status des Experimentes informiert. Diese Informationen müssen für das Bedienpersonal visualisiert werden, und dessen Anweisungen müssen im Rahmen des Messbetriebes entgegengenommen werden.

Der Name Slow Control rührt daher, dass sich die überwachten Parameter zeitlich nur langsam ändern, d.h. langsam im Vergleich zur Datenerfassung.

Das Slow Control System ist derart konzipiert, dass es eine übergeordnete Kontrollinstanz bildet. Das bedeutet unter anderem, dass es als Erstes gestartet und als Letztes beendet wird. Wenn sie ausfällt, wird das ganze Gebäude in einen „sicheren“ Status versetzt.

Um der Aufgabe eines sicheren und stabilen Experimentbetriebes gerecht zu werden, hat das Slow Control System „das letzte Wort“. Bei externen Befehlen - wie z.B. vom Bedienpersonal - muss das Slow Control System entscheiden, ob diese Befehle im Rahmen der festgelegten Sicherheitsbestimmungen zum gegenwärtigen Zeitpunkt ausführbar sind.

### 3.6.1 Hardware

Der so genannte FieldPC [FIELD05] ist die zentrale Instanz des Slow Control Systems. Er ist Industrie-PC, der auf einer x86-PC-Architektur basiert. Der FieldPC besitzt keinen Prozessorlüfter und auch keinen Gehäuselüfter. Stattdessen ist die gesamte Oberseite des Gehäuses mit Kühlrippen versehen und direkt auf dem Prozessor geschraubt.

Der FieldPC übernimmt mittels einer PROFIBUS-Schnittstelle und der entsprechenden Software (siehe Kap. 3.6.2) die Ansteuerung eines Busklemmensystems [BECK05] der Firma Beckhoff, das Sensor-, Aktuator-, Versorgungs- und System-Klemmen besitzt.

Außerdem steuert der FieldPC die Hochspannungsversorgung der Photomultiplier. Bei der Hochspannungsversorgung für die Photomultiplier handelt es sich um ein HV-Crate [CAEN] von der Firma CAEN, welches einen internen OPC-Server besitzt. Über diesen OPC-Server können z. B. Spannungswerte ausgelesen und gesteuert werden.

### 3.6.2 Software

Auch softwareseitig wurde auf Überflüssiges verzichtet. Der FieldPC wird mit einem auf das notwendigste Minimum beschränkten Betriebssystem (Windows NT 4.0) mit einem Microsoft Peer Web Service, einer Fernwartungssoftware (VNC) und der Software 4C [FIELD05] der Firma Softing ausgeliefert. Diese Software ist eine Entwicklungs- und Laufzeitumgebung und unterteilt sich in 4C Control, 4C Configuration, 4C Connectivity und 4C Console.

4C Control, das Laufzeitsystem, übernimmt das Abarbeiten des IEC 61131-3-Programmcodes auf dem FieldPC und organisiert die Kommunikation zur 4C Configuration.

4C Configuration ist das integrierte Programmiersystem nach IEC 61131-3. Auf den FieldPCs im Piere Auger Observatorium wird es nur benutzt, um die Programme zu übersetzen und an 4C Control zu übertragen.

4C Connectivity stellt offene standardisierte Schnittstellen und Funktionen bereit und übernimmt neben der PROFIBUS-Steuerung auch die Kommunikation mit dem HV-Crate über den OPC DA Client. Dazu bietet 4C Connectivity die Daten von 4C Control über einen OPC DA Server an.

4C Console ist ein Visualisierungs- und Steuerungstool, das die Daten der 4C Control als HTML-Seiten mit Hilfe des Microsoft Peer Web Service bereitgestellt. Diese Seiten müssen in ASP programmiert werden. Da die 4C Console HTML-Code erzeugt, welcher nur mit Microsoft Java Script darstellbar ist, kann die 4C Console nur im Zusammenspiel mit einem Internet Explorer der Firma Microsoft betrieben werden.

# 4 Analysephase

In diesem Kapitel wird zuerst der gegenwärtige Zustand diskutiert. Dabei wird auf die Teile eingegangen, die für diese Arbeit eine Rolle spielen. Danach werden die Probleme dargestellt und im letzten Teil die daraus resultierenden Anforderungen bestimmt.

## 4.1 Ist-Zustand

Das Pierre Auger Observatorium ist so organisiert, dass während des normalen Messbetriebs nur das CDAS mit Personal besetzt ist. Von hier aus werden alle Messeinrichtungen gesteuert, und hier laufen die gemessenen Experimentdaten zusammen.

In jedem FD-Gebäude gibt es pro Teleskop einen „Mirror PC“ für die Kommunikation mit dem Teleskopen, einen „Eye PC“ für die Datensammlung und die Kommunikation zum CDAS und einen FieldPC für die Steuerung der Messeinrichtungen.

Die Hard- und Software in den FD-Gebäuden sind für den stabilen Messbetrieb ausgelegt. Deshalb ist in diesen Gebäuden jedes wichtige elektrische Gerät des Slow Control System - wie z. B. der FieldPC und die Netzwerkkomponenten - durch eine USV vor Spannungsschwankungen und -ausfällen geschützt.

Die Linux PCs im CDAS und in den FD-Gebäuden verwenden die Distribution von SUSE in der Version 9.1.

### 4.1.1 Netzwerkinfrastruktur

Das Netzwerk im Pierre Auger Experiment benutzt das Internetprotokoll TCP/IP. Die einzelnen Gebäude, wie das CDAS und die 4 FD-Gebäude, sind in einzelne Subnetze aufgeteilt. Diese Subnetze liegen im privaten C-Bereich, also IP-Adressen im 192.168.x.x-Bereich. Zwischen den einzelnen Subnetzen sorgen Router für den Transport der IP-Pakete (siehe Abbildung 4.1).

Für die weiten Strecken zwischen CDAS und den FD-Gebäuden wurden Richtfunkstrecken mit einer Bandbreite von 2 MBit/s eingesetzt. Bei diesen Verbindungen muss beachtet werden, dass es durch äußere Parameter - wie z. B. schlechtes Wetter und Spannungsschwankungen - zu kurzen Unterbrechungen kommen kann.

In den Teleskopgebäuden existiert ein 10/100MBit/s Ethernetnetzwerk mit Twisted-Pair- bzw. Lichtwellenleiter-Verkabelung.

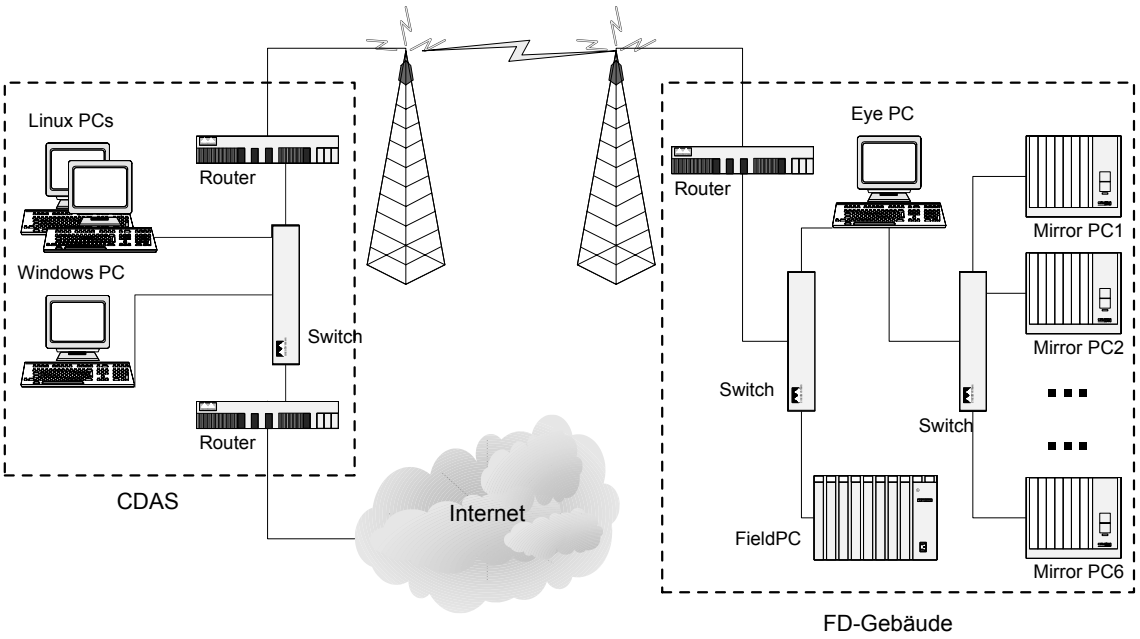


**Abbildung 4.1 Netzwerkschema des Pierre Auger Experiments (reduziert auf CDAS und FD-Gebäude).**

### 4.1.2 Kontrolldatenfluss

Abbildung 4.2 zeigt den Kontrolldatenfluss beim Pierre Auger Experiment. Die Slow Control Systeme werden mit Hilfe der 4C Console und eines Internetbrowsers auf einem Windows PC im CDAS angebunden. Bei dieser Anbindung wird pro Gebäude ein Browser benutzt.

Die Datenerfassung (DAQ) in den FD-Gebäuden kommuniziert über TCP/IP mit einer grafischen Benutzeroberfläche im CDAS.

Es hat bereits Versuche gegeben, eine Verbindung zwischen dem Slow Control System und der DAQ über OPC DA mit Hilfe eines Linux OPC DA Clients zu implementieren. Diese Verbindungen erwiesen sich jedoch als nicht stabil und unpraktikabel. Deshalb ist momentan der Operator im CDAS die einzige Verbindung zwischen dem Slow Control System und der DAQ.

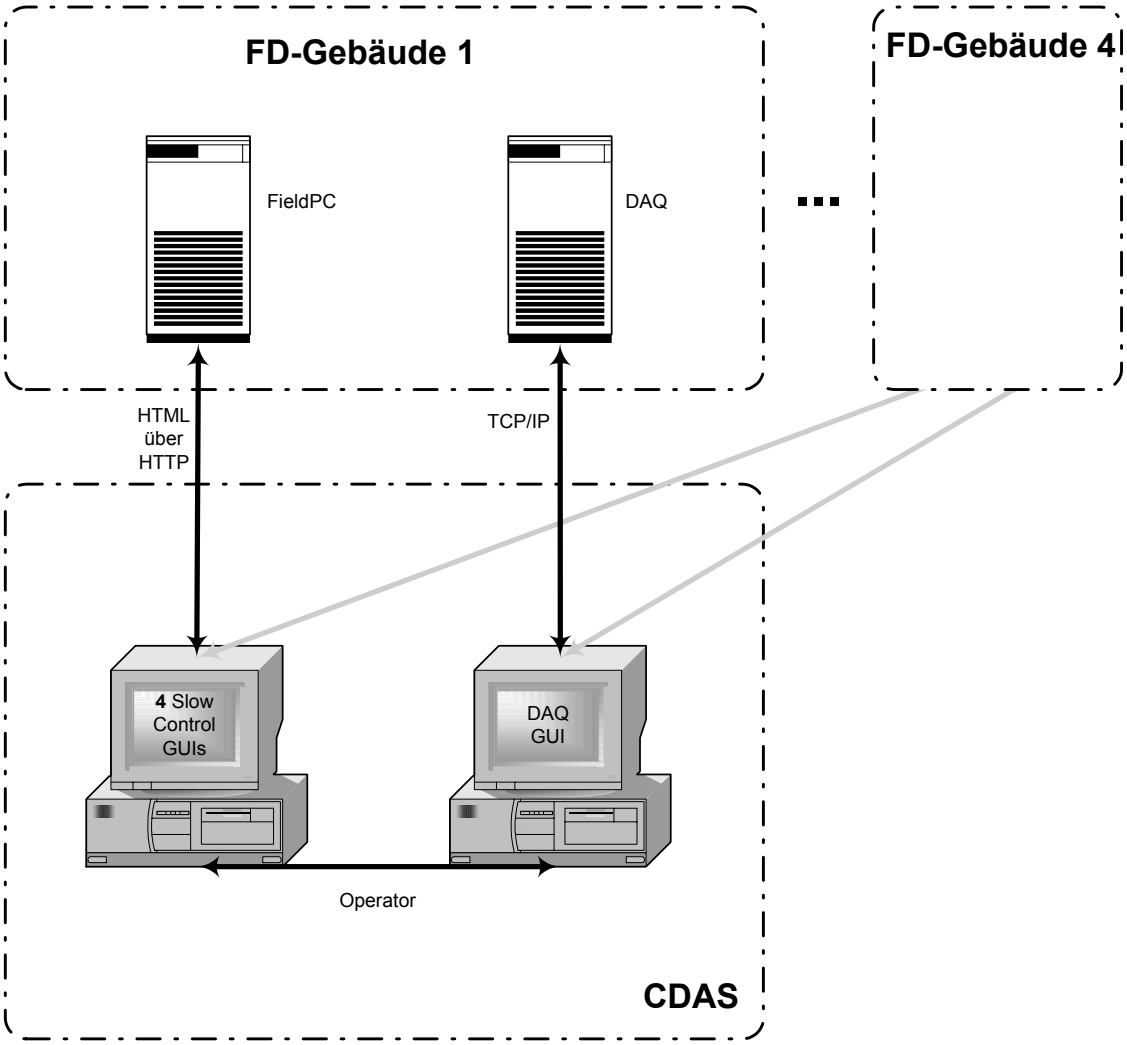


**Abbildung 4.2 Kontrolldatenfluss beim Pierre Auger Experiment.**

### 4.1.3 Der FieldPC

Wie in Kap. 3.6.1 beschrieben, ist der FieldPC die zentrale Instanz des Slow Control Systems. Diese FieldPCs sind mit dem Betriebssystem Windows NT 4.0 der Firma Microsoft, einer Fernwartungssoftware VNC und der Anwendungssoftware 4C der Firma Softing installiert.

Die Ressourcen des FieldPCs sind so ausgelegt, dass sie für die installierten Programme ausreichend sind. Die 128MB RAM Speicher, von denen im Normalbetrieb zwischen 90 und 100 MB gebraucht werden, und die CPU, die mit 750 MHz nicht zu der neuesten Generation gehört, sind nicht veränderbar. Damit ein stabiler Betrieb des Slow Control System sichergestellt werden kann, können Programme mit größerem Ressourcenverbrauch auf den FieldPCs nicht installiert werden.

## 4.2 Problemstellung

Aus der Ist-Zustands-Analyse ergeben sich folgende Verbesserungen: Wegen der fehlenden Verbindung zwischen dem Slow Control System und der DAQ können auch Messungen gestartet werden, obwohl die Messeinrichtung noch nicht betriebsbereit ist. Fehlerzustände führen nicht zu einem Messabbruch.

Zur Überwachung der FD-Gebäude braucht man mehrere geöffnete Internetbrowser. Wenn ein Browser nicht sichtbar ist, können wichtige Ereignisse übersehen werden.

Die HTML-Seiten der 4C Console werden zyklisch von den FieldPCs in den FD-Gebäuden angefordert. Dieses geschieht auch, wenn sich nichts geändert hat.
Die Operatoren im CDAS sind nicht geschult im Umgang mit Windows Betriebssystemen, und im CDAS soll aus Wartbarkeitsgründen dauerhaft kein Rechner mit einem Windows Betriebsystem betrieben werden.

Die Daten der Slow Control Systeme - wie z. B. Wetterdaten, Zustände und Steuerbefehle - werden nur in Log-Dateien lokal auf dem FieldPC abgelegt. Diese Dateien werden zyklisch auf einem Rechner im CDAS gesichert.

## 4.3 Resultierende Anforderungen

Aus dem oben aufgezeigten Ist-Zustand und seinen Problemen ergeben sich Anforderungen für einen sichereren und stabileren Betrieb.

Um eine stabile Kommunikation zwischen dem Slow Control System und der DAQ zu implementieren, braucht man eine neue Schnittstelle. Eine Möglichkeit wäre die Ausnutzung der OPC XML-DA Spezifikation (siehe Kap. 3.5.3).

### 4.3.1 OPC XML-DA Gateway

Um dieses Protokoll benutzen zu können, müsste ein Gateway auf den FieldPCs installiert werden, der auf die OPC DA Schnittstelle zugreift und eine OPC XML-DA anbietet. Dieser Gateway muss jedoch mit den begrenzten Ressourcen des FieldPCs auskommen. Dazu sollte der Gateway Daten puffern können, um auch bei Netzwerkstörungen eine konsistente Datenarchivierung zu ermöglichen. Die Schreib- und Leserechte auf den Daten der Elemente des OPC DA Servers sollten parametrisierbar sein, und für Schreibrechte sollte eine Authentifizierung existieren.

Falls es keine kommerziellen Produkte gibt, wird eine Eigenentwicklung erfolgen.

### 4.3.2 OPC XML-DA Client Komponente

Um mit dem Gateway kommunizieren zu können, braucht man eine OPC XML-DA Client Komponente (siehe Abbildung 4.3). Diese sollte sich einfach in bestehende C++ Programme einbinden lassen.

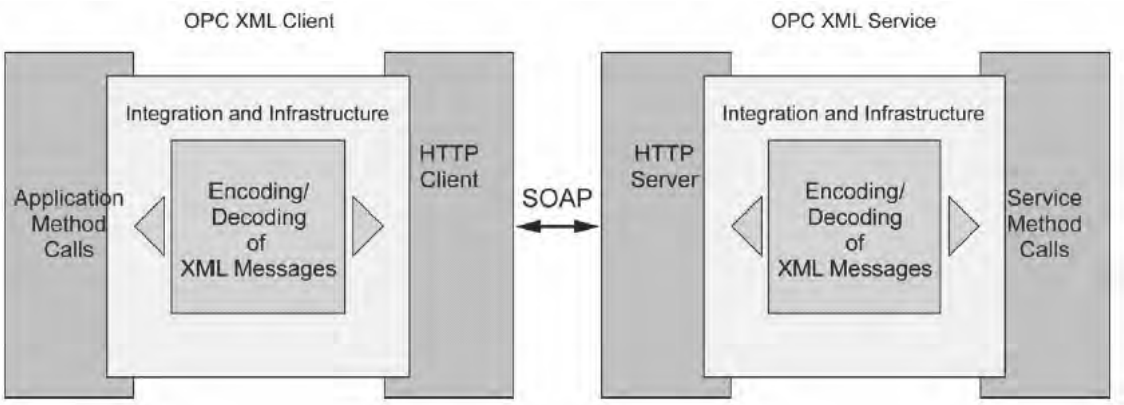


**Abbildung 4.3 Schematischer Datenfluss und Datenumsetzung zwischen Client und Server.**

Da das DAQ-Programm nicht „Thread-sicher“ ist, ergibt sich eine weitere Anforderung: Eine erweiterte Komponente sollte zur Verfügung stehen, die eigenständig läuft und die Daten mit der DAQ über Shared Memory austauscht.

Das OPC XML-DA Protokoll ist durch das HTTP-Protokoll verbindungslos. Trotzdem sollte auf die Netzwerkumgebung, besonders das Timeout-Verhalten von TCP/IP, Rücksicht genommen werden.

Diese Komponente soll unter Windows und Linux einsetzbar sein.

### 4.3.3 Visualisierungs- und Steuerungssystem

Damit der Windows-Rechner, auf dem die HTML-Seiten der 4C Consolen angezeigt werden, nicht mehr im CDAS gebraucht wird, sollte ein neues Visualisierungs- und Steuerungssystem (SCADA System) entwickelt werden.

Diese Software soll im CDAS auf einem dafür bestimmten Rechner laufen und über OPC XML-DA mit dem Gateway auf den FieldPCs in den FD-Gebäuden kommunizieren. Die empfangenen Daten sollen zwischengepuffert werden, damit eine Clientanfrage nicht jedes Mal eine Kommunikation mit dem Gateway erfordert.

Diese Software sollte ähnliche HTML-Seiten wie die 4C Console erzeugen, nur dass diese auf allen gängigen Internetbrowsern - wie z. B. Opera und Mozilla - anzeigbar sind.

In einer HTML-Seite sollten Daten von mehreren Slow Control Systemen angezeigt werden können, damit nur ein Browser geöffnet sein muss, um alle FD-Gebäude steuern und überwachen zu können. Außerdem sollten mehrere User - wie z. B. Entwickler im Rest der Welt - gleichzeitig auf das System zugreifen können.

Die Seiten sollten keine Logik für die Steuerung der Slow Control Systeme beinhalten, da diese selbst für den sicheren Ablauf sorgen. Die Sicherheit ist nicht gefährdet, wenn falsche Steuerungsbefehle zum Slow Control System gesendet werden.

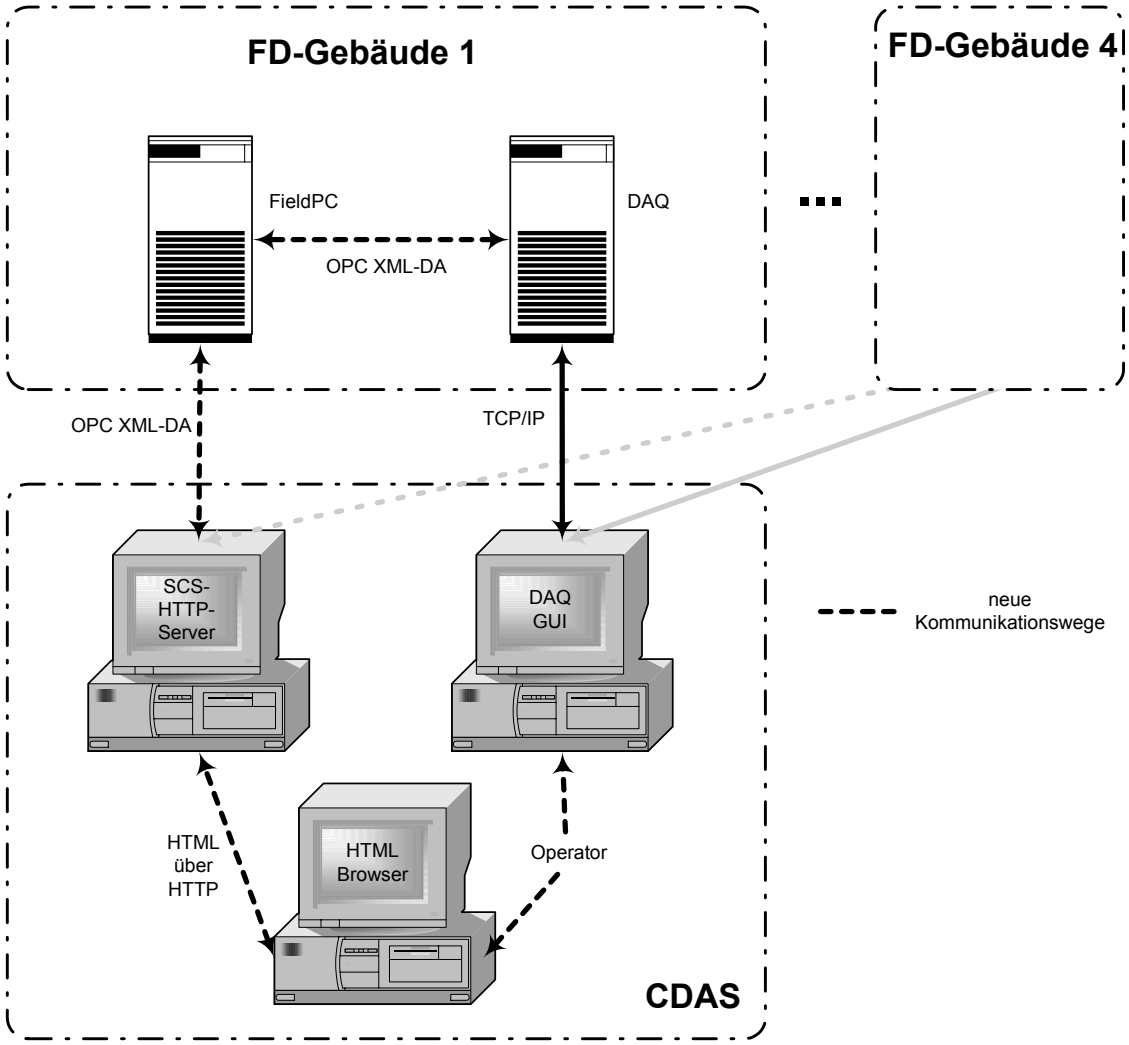


**Abbildung 4.4 Geplanter Kontrolldatenfluss im Pierre Auger Experiment.**

Für Arbeiten vor Ort in den FD-Gebäuden soll nicht der Umweg über die Richtfunkstrecken genommen werden. Dafür soll diese Software auf dem FieldPC oder einem anderen Rechner im FD-Gebäude betrieben werden.

Des Weiteren sollten die anzuzeigenden Seiten schnell und einfach an neue Bedürfnisse anzupassen sein, da sich das Projekt noch in der Aufbauphase befindet und die Anforderungen bzw. auch die Programme des Slow Control Systems sich noch ändern können.

# 5 Designphase

In dieser Phase soll geplant werden, welche Komponenten benutzt und wie sie einzusetzen sind. Bestehende Komponenten, die schon ihre Stabilität, Sicherheit und Robustheit im Einsatz erwiesen haben, werden gegenüber Eigenentwicklungen bevorzugt.
In dieser Phase wird nur das Design der einzelnen zu verwendenden Komponenten und deren Umsetzung diskutiert.

## 5.1 OPC XML-DA Gateway Produkte

### 5.1.1 Tests von OPC XML-DA Gateway Produkten

Bei Recherchen im Internet zu Beginn dieser Arbeit konnte man nur zwei OPC XML-DA Gateways finden, die das Protokoll OPC DA nach OPC XML-DA umsetzten.

### 5.1.2 OPC XML-DA Bridge Server Side Gateway

OPC XML-DA Bridge Server Side Gateway [TECHN] der Firma Technosoft ist der erste Testkandidat gewesen. Bei Installationsversuchen auf den FieldPC stellte sich heraus, das der Gateway sich zwar installieren lässt, aber für den Betrieb nicht nur das .NET Framework, sondern auch das ASP.NET, welches als Systemanforderung Windows 2000 oder höher hat, benötigt. Da das Betriebssystem auf dem FieldPC festgelegt ist, ist dieser Gateway nicht einsetzbar.

### 5.1.3 dOPC XGate

„dOPC XGate“ [KASSL] in der Version 1.31 von der Firma Kassl ließ sich auf dem FieldPC installieren. Dieser Gateway liefert auch erst einmal den gewünschten Erfolg. Doch bei einfachen Tests mit verschiedenen OPC XML-DA Test-Clients kommt es immer wieder zu spontanen und nicht reproduzierbaren Abstürzen des Gateways. Da es noch keine weitere Version gibt, die dieses Problem behebt, kann dieser Gateway ebenfalls nicht eingesetzt werden.

## 5.2 OPC XML-DA Gateway Entwicklung

Da für die Anforderungen keine kommerzielle Lösung existiert, muss dieser Gateway entworfen und umgesetzt werden.

Für einen OPC XML-DA Gateway werden zwei Komponenten gebraucht (siehe Abbildung 5.1). Den SOAP-Teil, der die Messages der OPC XML-DA Spezifikation entgegen nimmt, und den OPC DA Teil, der für diese Messages Methoden bereitstellt und damit definiert auf den OPC DA Server zugreift.

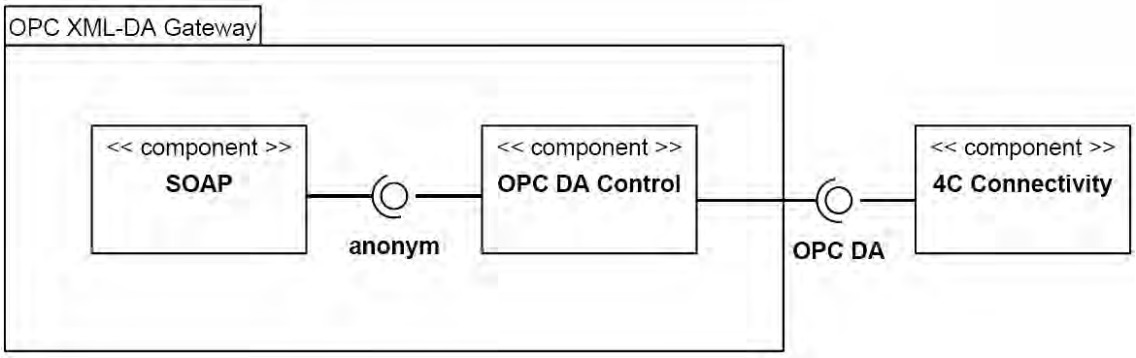


**Abbildung 5.1 Komponentendiagramm des OPC XML-DA Gateway.**

### 5.2.1 OPC XML-DA Komponente

#### 5.2.1.1 SOAP-Framework Evaluierung

Um den Entwurf und die Implementierung zu vereinfachen, wäre ein fertiges SOAP-Framework, das die WSDL-Datei der OPC XML-DA Spezifikation verarbeiten kann, von Vorteil.

Auf dem Markt gibt es zwar eine Menge von SOAP-Frameworks; die meisten unterstützen jedoch nicht C++.

Ein weiteres Kriterium ist, dass das Framework unter Linux und Windows einsetzbar ist, damit man sich nur in ein Framework einarbeiten muss. Dadurch wird die Umsetzung und die Wartung vereinfacht. Außerdem soll das Framework so wenig wie möglich Ressourcen benutzen, damit diese Komponente auch auf den FieldPCs lauffähig ist.

#### 5.2.1.1.1 gSOAP

Ein schlankes und einfaches Framework ist gSOAP [GSOAP]. Es ist ein Open Source Framework. Da gSOAP den typlosen Datentyp „Value“ der WSDL-Datei nicht umsetzen kann, dieser aber sehr wichtig ist, kann dieses Framework nicht eingesetzt werden.

#### 5.2.1.1.2 AXIS C++

Ein weiteres Open Source Projekt ist AXIS C++ von der Apache Software Foundation [AXIS]. Dieses Framework kann nicht verwendet werden, da wegen des großen Ressourcenverbrauches der Speicher des FieldPCs nicht ausreichend ist.

#### 5.2.1.1.3 Qt

Zu Beginn dieser Arbeit wurde von Trolltech eine neue SOAP Library für Qt [QT] angekündigt. Da sie zu diesem Zeitpunkt noch nicht verfügbar war, wurde sie nicht weiter berücksichtigt.

### 5.2.1.2 SOAP-OPC XML-DA Server Komponente

Da kein den Anforderungen entsprechendes Framework für SOAP existiert, muss ein eigenes entwickelt werden. Weil ein vollständiges SOAP-Framework den zeitlichen Umfang dieser Arbeit überschreiten würde, wird nun eine vereinfachte Umsetzung von SOAP entworfen.

Die OPC XML-DA Schnittstelle verwendet „SOAP over HTTP“. Darum wird diese Komponente aus zwei Komponenten, einem HTTP-Server und dem OPC XML-DA zusammengesetzt (siehe Abbildung 5.2).

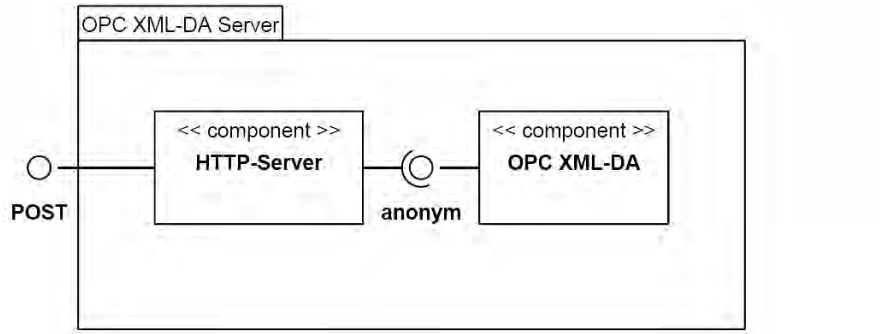


**Abbildung 5.2 Komponentendiagramm der SOAP-OPC XML-DA Server Komponenten.**

#### 5.2.1.2.1 HTTP Server Komponente

Die HTTP-Server Komponente muss nur die POST-Methode von HTTP unterstützen, da diese nur für „SOAP over HTTP“ verwendet wird.

Für diese Komponente wird eine Eigenentwicklung eingesetzt, die bereits ausreichende Stabilität und Robustheit durch ihren mehrfachen Einsatz bewiesen hat [BARKH].

#### 5.2.1.2.2 OPC XML-DA Komponente

Das mit der POST-Methode übertragene SOAP-Payload ist im Grunde ein XML-File mit unterschiedlichen Ausprägungen. Es gibt pro Nachricht ein XML-File für die Anfrage und eines für die Antwort. Da OPC XML-DA nur acht Methoden besitzt, wird ein Automatismus, der die Klassenstruktur aus der WSDL-Datei generiert, keinen Zeitvorteil bringen.

Darum wurde hier eine einfache Umwandlung vorgenommen. Die Strukturen werden eins zu eins umgesetzt (siehe Abbildung 5.3). Aus jedem Element der SOAP-Nachricht in der WSDL-Datei wird eine Klasse in C++. Jedes Attribut eines Elementes wird zu einer Membervariablen der C++ Klasse. Wenn ein Element in einem anderen Element enthalten ist, so wird dieses auch zur Membervariable, wenn es genau einmal vorkommt, oder es wird ein Vektor seiner Klasse, wenn dieses Element mehrfach vorkommen kann.

Das Element „Value“ weicht von dieser Regel ab. Es ist in der WSDL-Datei ein Element ohne Typdefinition. Der Typ wird als Attribut „xsi:type“ mitgeliefert bzw. muss mitgeliefert werden. Darum soll diese Klasse so umgesetzt werden, dass sie ein Element mit einem Attribut und einem Wert ist. Zur Vereinfachung werden Array-Typen nicht berücksichtigt.

Für das Parsen des XML-File wird die Xerces-C Library [XERCE] von der Apache Software Foundation benutzt, da diese im Pierre Auger Projekt schon im Einsatz ist.

Die Umwandlung findet immer über die DOM[7]-Struktur statt. Das heißt, dass aus einer SOAP-Nachricht eine DOM-Struktur und diese dann die Klassenstruktur wird.

[7] Abkürzung für Document Object Model.

Umgekehrt wird aus der Klassenstruktur eine DOM-Struktur und daraus eine SOAP-Nachricht.

```
   ...
<s:element name="Subscribe">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="Options"
                                      type="s0:RequestOptions" />
      <s:element minOccurs="0" maxOccurs="1" name="ItemList"
                             type="s0:SubscribeRequestItemList" />
    </s:sequence>
    <s:attribute name="ReturnValuesOnReply" type="s:boolean"
                                                 use="required" />
    <s:attribute default="0" name="SubscriptionPingRate"
                                                   type="s:int" />
  </s:complexType>
</s:element>
   ...
```

```
   ...
class MESSAGE_SUBSCRIBE : public MESSAGE
{
public:
   bool                          create_by_node(DOMNode * _node);
   bool                          serialize(DOMElement & _node,
                                    DOMDocument & _doc) const;
   ...
// Elemente:
   REQUEST_OPTIONS               m_options;
   SUBSCRIBE_REQUEST_ITEM_LIST   m_item_list;

// Attribute:
   bool                          m_return_values_on_reply;
   int                           m_subscription_ping_rate;
   ...
```

**Abbildung 5.3 Beispiel für die Umsetzung; Ausschnitt aus der WSDL-Datei (oben) und den daraus entstandenen C++-Code (unten).**

Dieser Zwischenschritt verbraucht zwar mehr Speicher, als wenn er weggelassen würde, jedoch sind die XML-Nachrichten nicht sehr groß (wesentlich kleiner als 1 MB). Dafür kann einfacher zu einer anderen XML-Library (wie z. B. die von Qt) gewechselt werden. Im Pierre Auger Projekt wird momentan darüber nachgedacht, dass man Xerces-C durch Qt-XML ersetzt, da Qt für die grafischen Oberflächen schon eingesetzt wird.

Für die zwei Umwandlungen hat jede Klasse die Methode „serialize“, also für die Umsetzung der Klassenstruktur in die DOM-Struktur und „create_by_node“ für die

andere Richtung. Diese Methoden werden rekursiv auch bei den enthaltenen Klassen aufgerufen.

### 5.2.2 OPC DA Komponente

Die OPC DA Komponente hat auf der einen Seite eine Schnittstelle zu dem OPC Server, die durch DCOM definiert ist, auf der anderen Seite stellt die Komponente eine Schnittstelle bereit, die die in der OPC XML-DA Komponente erzeugten Strukturen entgegennehmen kann. Damit liegt die eigentliche Logik und die Umsetzung zwischen den OPC DA und dem OPC XML-DA Strukturen in dieser Komponente.

Diese Komponente übernimmt dabei auch das Zwischenspeichern der Daten der Elemente, die durch eine Subscription angelegt wurden. Des Weiteren steuert sie die Zugriffsrechte auf die Elemente.

## 5.3 OPC XML-DA Client Komponente

Die OPC XML-DA Client Komponente wird, wie der Gateway, aus mehreren Komponenten zusammengebaut. Die Hauptkomponenten sind ein HTTP-Client, eine OPC XML-DA Komponente und eine Client-Steuerungskomponente (siehe Abbildung 5.4).

Es gibt auch OPC XML-DA Client Toolkits von mehreren Anbietern. Diese Toolkits werden nicht eingesetzt, da der Großteil der notwendigen Komponenten bereits für den Gateway entwickelt wurden oder schon fertig vorhanden sind. Außerdem müssten diese Toolkits auch noch evaluiert werden, um ihre Einsetzbarkeit zu testen.

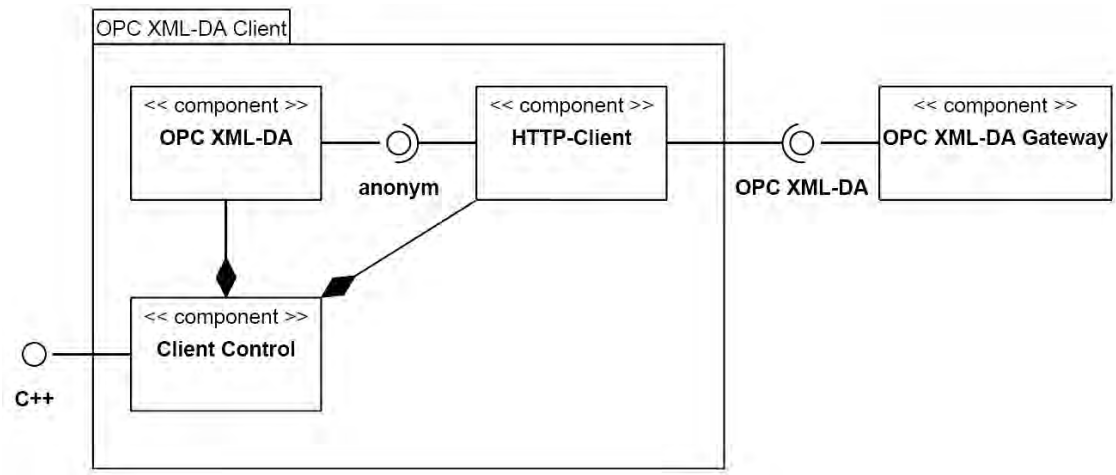


**Abbildung 5.4 Komponentendiagramm OPC XML-DA Client.**

### 5.3.1 HTTP-Client

Die HTTP-Client Komponente muss genauso wie der HTTP-Server (siehe Kap. 5.2.1.2.1) nur die POST-Methode unterstützen.

Für diese Komponente wird eine Eigenentwicklung eingesetzt, die schon ausreichende Stabilität und Robustheit durch ihren mehrfachen Einsatz bewiesen hat [BARKH].

### 5.3.2 OPC XML-DA Komponente

Als OPC XML-DA Komponente wird die gleiche Komponente, die schon für den OPC XML-DA Gateway (siehe Kap. 5.2.1.2.2) entwickelt wurde, verwendet.

### 5.3.3 Client-Steuerungs Komponente

Damit der HTTP-Client und die OPC XML-DA Komponente miteinander zusammenarbeiten gibt es die Client-Steuerungs Komponente. Sie wird mit Hilfe einer Datei konfiguriert und steuert auf der einen Seite die Kommunikation zu dem OPC XML-DA Gateway. Auf der anderen Seite gibt es eine C++ Schnittstelle, mit der auf die konfigurierten Variablen des OPC DA Servers über den Gateway zugegriffen wird.

Die Daten der Elemente werden über eine Subscription vom Gateway abgerufen und in dieser Komponente zwischengespeichert. Damit muss nicht bei jedem Zugriff auf die C++ Schnittstelle ein Zugriff auf den Gateway stattfinden, und es werden nur Daten vom Gateway übertragen, die sich auch geändert haben.

## 5.4 OPC XML-DA Client Komponentenerweiterung

Für die DAQ-Anwendung wird noch eine Erweiterung gebraucht, da es in der Client-Steuerung und der HTTP-Client Komponente Threads gibt. Diese Komponente beinhaltet den OPC XML-DA Client und eine Shared Memory Komponente und läuft als eigenständiger Prozess (siehe Abbildung 5.5).

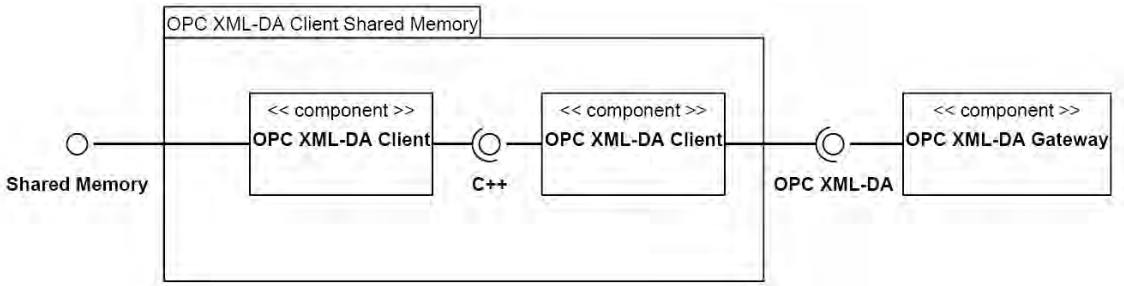


**Abbildung 5.5 Komponentendiagramm der Erweiterung des OPC XML-DA Clients.**

Diese stellt eine Schnittstelle für das Lesen und Schreiben von Variablen in das Shared Memory bereit. Diese Komponente ist ein Server für die Daten, die vom Gateway kommen, und ein Client für die Daten die zum Gateway geschickt werden. Jeder Teil hat seinen eigenen Bereich im Shared Memory, der durch eine Struktur festgelegt ist.

## 5.5 Visualisierungs- und Steuerungssystem (SCADA)

Das Visualisierungs- und Steuerungssystem, des SCS-HTTP-Server, baut auf dem OPC XML-DA Client und der HTTP-Server Komponente auf (siehe Abbildung 5.6). Bei der HTTP-Server Komponente wird diesmal die GET-Methode gebraucht. Hinzu kommt eine SCS-Control Komponente.

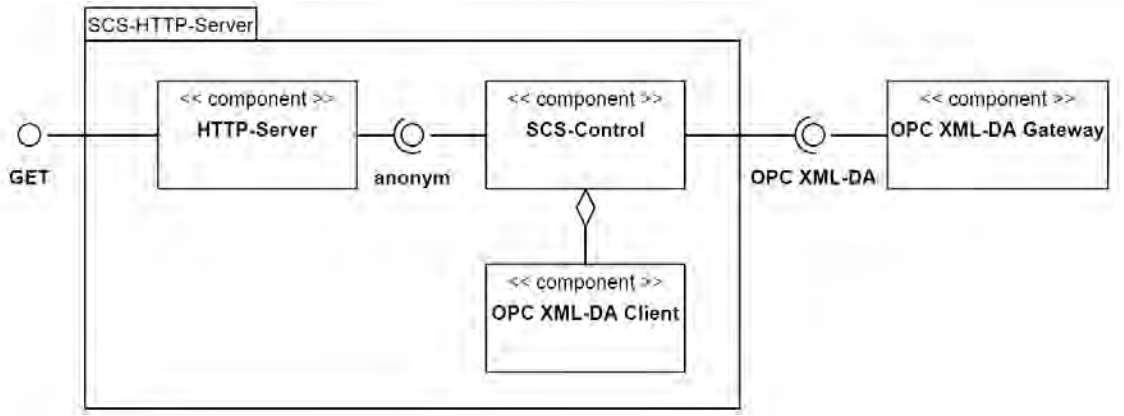


**Abbildung 5.6 Zusammenarbeit der Komponenten beim SCS-HTTP-Server.**

Die SCS-Control Komponente wird durch eine Datei konfiguriert. Je nachdem wie sie konfiguriert wird, hat sie eine oder mehrere Instanzen der OPC XML-DA Client Komponente. Damit kann diese Komponente auf die Variablen von mehreren OPC DA Servern zugreifen.

Die SCS-HTTP-Server Komponente nimmt die GET-Anfragen eines HTTP-Clients entgegen. Vom Root-Verzeichnis dieses Servers gibt es zwei virtuelle Verzeichnisse, die unterschiedlich behandelt werden.

Die aus dem Verzeichnis „static“ angeforderten Dateien werden unverändert als HTTP-Anwort zurückgeliefert. Dieses Verzeichnis ist für Bilder und statische HTML-Seiten vorgesehen, darum werden sie mit einer großen Ablaufzeit übermittelt, damit der Browser sie nicht immer neu laden muss.

Die aus dem Verzeichnis „dyn“ angeforderten Dateien werden geparst, und jeder String mit dem Format „$$Variablenname$$“ wird durch den Variablenwert ersetzt. Diese Datei wird mit einer Ablaufzeit von 0 übertragen, damit sie nicht vom Browser gepuffert wird.

Damit lassen sich die vorhandenen Seiten der 4C Console einfach umsetzen. Im Gegensatz zu der 4C Console kann der SCS-HTTP-Server auf mehrere Slow Control Systeme zugreifen. Damit lassen sich Daten von mehreren Systemen in einer HTML-Seite darstellen.

# 6 Implementierungsphase

## 6.1 Orgware[8]

Als Entwicklungsumgebung wurde Visual Studio 6.0 und für die Versionsverwaltung SourceSafe von der Firma Microsoft verwendet.

Die Teile der Software, die auch unter Linux benutzt werden, wurden dort mit dem GCC-Compiler der Version 3.3.3 kompiliert und gelinkt. Für beides wurde ein Shell-Script geschrieben. Wenn unter Linux ein Debugger nötig war, wurde der Display Data Debugger (DDD) verwendet.

Außerdem wurden zum Testen CppUnit, der XMLDA.NET Testclient der Firma Technosoft, der dOPC Explorer der Firma Kassl und Rational Purify for Windows von der Firma IBM verwendet.

Für die Entwicklung und die Tests wurden PCs mit x86-Architektur mit den Betriebssystemen Microsoft Windows XP bzw. SUSE Linux 9.1 und ein FieldPC genutzt, die über ein geswitchtes Ethernet miteinander verbunden waren.

## 6.2 Implementierung

Die gesamte Implementierung wird in C++ durchgeführt und es werden nur Komponenten benutzt, die unter Windows und Linux zu verwenden sind. Deshalb wurde soweit wie möglich versucht, nur Datentypen zu verwenden, die es auf beiden Systemen gibt. Eine Ausnahme wurde bei der OPC DA Komponente gemacht, weil OPC DA den Datentyp VARIANT verwendet, der nicht unter Linux existiert. Dies konnte so umgesetzt werden, da die OPC DA Komponente nur im Gateway auf den FieldPCs betrieben wird.

Der OPC XML-DA Gatway, der OPC XML-DA Client, die OPC XML-DA Client Erweiterung und der SCS-HTTP-Server wurden unter Windows entwickelt. Nachdem dort jede Komponente vollständig funktionierte, wurden diejenigen, die unter Linux betrieben werden sollten, dort auch kompiliert und aus den Sourcen ausführbare Dateien erstellt.

[8] Das Kunstwort „Orgware“ beschreibt die Umgebung, mit welcher entwickelt wird.

Für den SCS-HTTP-Server wurden HTML-Seiten entwickelt, die vergleichbar mit denen der 4C Console sind.

## 6.3 Testmethoden

Effizientes Testen heißt, möglichst zeitnah zur Programmierung, automatisiert und damit wiederholbar, Fehlerfreiheit zu beweisen. Programmtests sind sehr nützlich, um das Vorhandensein von Fehlern zu zeigen, nie aber ihre Abwesenheit [DIJKS72].
Bei Implementierungen können immer Fehler entstehen. Deshalb wurden auch bei dieser Unit-, Funktions-, Performanz- und Last-Tests durchgeführt. Zusätzlich werden auch noch die Programme auf Memory Leaks (Speicherfresser) untersucht.

### 6.3.1 Unit-Test

Beim Unit-Test werden die Funktionen eines Teils des Systems, wie Klassen und Komponenten, isoliert getestet. Dieser Begriff stammt ursprünglich aus der vorobjektorientierten Ära und beschreibt, dass sich einzelne Tests nicht auf das Gesamtsystem, sondern auf einzelne Einheiten (Units) des Systems konzentrieren. Sie fallen in die Kategorie der Whitebox-Tests.

In dieser Arbeit wurde CppUnit, ein Open Source Projekt, benutzt. CppUnit ist ein Unit-Test-Tool und dient dem Testen von isolierten Programmeinheiten wie einzelnen Methoden, Klassen und Komponenten. Es setzt an den öffentlichen Schnittstellen dieser Einheiten an. Mit CppUnit werden wiederholbare Testfälle, also Testdaten mit den erwarteten Ergebnissen, erstellt. Diverse gesammelte Tests können als Suiten zusammengestellt werden. Grenzen von CppUnit sind Tests graphischer Benutzeroberflächen, nebenläufiger Codes und mehrschichtiger Architektur. Der Erfolg oder Nicht-Erfolg der Tests wird in der Console ausgegeben. Bei Nicht-Erfolg wird auf den entsprechenden Testfall verwiesen.

Die Vorteile dieser Tests sind, dass der Pool der Testfälle mit der Zeit wächst, die Tests immer gleich ablaufen, das Refactoring[9] der Komponenten einfacher wird und der Tester nur Zeit für den Start und die Auswertung aufwenden muss, nicht aber die

[9] Umgestalten (eben das re-factoring) von Software-Komponenten ohne eine Änderung an der öffentlichen Schnittstelle.

des Testlaufs. Deshalb wurde bei der Arbeit versucht, den lauffähigen Code direkt mit den bestehenden und neuen Testfällen zu testen.

### 6.3.2 Funktions-Test

Beim Funktions-Test werden Teile oder das Gesamtsystem gegenüber der Spezifikation überprüft. Funktions-Tests werden in der Regel vom Testteam durchgeführt, welches die Testergebnisse protokolliert. Das Testteam leitet die Ergebnisse an das Entwicklungsteam weiter, welches dann analysiert, warum etwas nicht funktioniert, und die Fehler behebt. Funktions-Tests fallen in die Kategorie der Blackbox-Tests.

Funktions-Tests am OPC XML-DA Gateway wurden mit Hilfe des XMLDA.NET Testclient und des dOPC Explorer durchgeführt. Dabei wurde getestet, ob der OPC XML-DA Gateway mit der OPC XML-DA Spezifaktion 1.01 übereinstimmt. Besonders geachtet wurde auf die Wandlung der Datentypen von OPC DA nach OPC XML-DA und umgekehrt sowie auf die Umsetzung der acht Methoden.

Der OPC XML-DA Client wurde gegen einen von der OPC Foundation bereitgestellten OPC XML-DA Server, der jeden Datentypen und jede Methode der Spezifikation bereitstellt, getestet. Zusätzlich wurde der Client gegen den Gateway getestet.

Zum Testen des SCS-HTTP-Servers wurden ASP-Seiten der 4C Console in HTML-Seiten für den SCS-HTTP-Server umgewandelt. Dann wurde verglichen, ob das Verhalten von dem SCS-HTTP-Server mit dem der 4C Console übereinstimmt. Hier wurde besonders auf die Anzeige von Werten und die Übertragung der Steuerungsbefehle geachtet.

### 6.3.3 Performanz- und Last-Test

Allgemein wird durch Last-Tests das Verhalten der Anwendung bei mehr als einem Benutzer simuliert; die Konzentration liegt auf der Erfüllung bestimmter nicht funktionaler Anforderungen, wie geforderte Antwortzeiten und maximale Nutzerzahlen.

Beim Performanz-Test wird die Anwendung mit der später in der Produktion erwarteten Anzahl von Benutzern unter Last gesetzt. Diese Tests sind in einer produktionsnahen Umgebung (Hardware, Betriebsystem, Software, etc) durchzuführen, da das Ergebnis sonst keine verlässliche Aussage erlaubt.

Für diese Tests wurde der Gateway auf dem FieldPC installiert. Dann wurden von dem OPC XML-DA Client auf einem anderen PC die Read- bzw. die SubscriptionPolledRefresh-Methode des Gateways aufgerufen. Dabei wurde die Zeit gemessen, die der Client vom Start der Anfrage bis zur Bereitstellung der Antwortdaten brauchte. Diese Messung wurde für eine unterschiedliche Anzahl von Elementen gemacht, und pro Elementenanzahl wurde der Durchschnittswert aus 50 Messungen berechnet und in einem Diagramm aufgetragen (siehe Abbildung 6.1).

Bei der „Read"-Methode war die CPU-Auslastung des FieldPCs schon mit ca. 200 Variablen auf 100%. Bei der „Subscription"-Methode fing diese Auslastung erst bei ca. 1000 Variablen an. Die Grundlast durch das 4C System beträgt dabei schon 50%.

Der Speicherbedarf ist bei den Tests immer unter den vorhandenen 128 MB geblieben, und es musste kein Speicher ausgelagert werden, was den Betrieb des Slow Control Systems stören könnte.

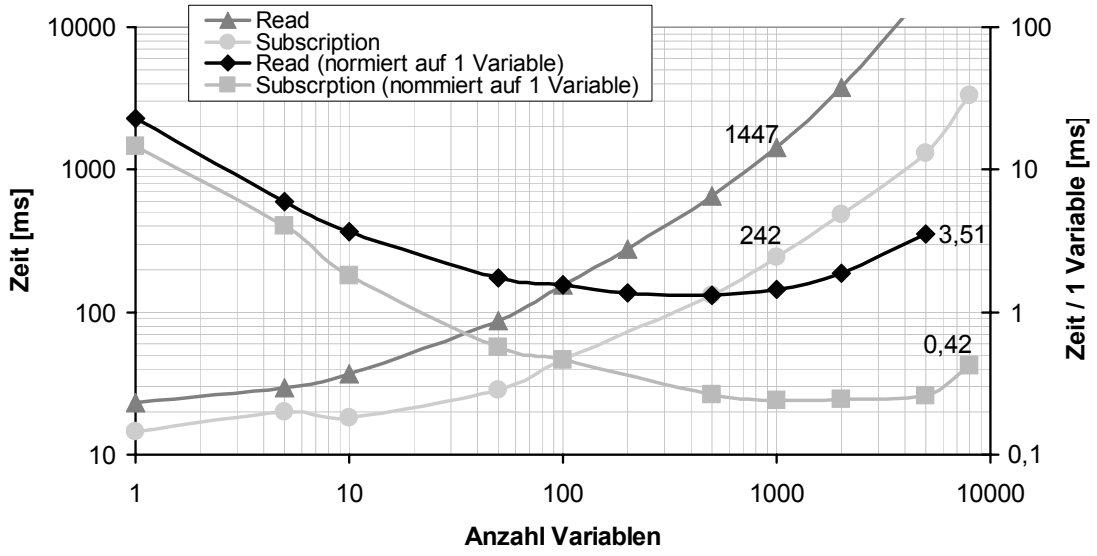


**Abbildung 6.1 Vergleich von der „Read"- und der „Subscription"-Methode (Linien dienen nur zur Führung der Augen).**

Der SCS-HTTP-Server wurde mit mehreren Browsern getestet. Bei zehn geöffneten Browsern, die im Sekundentakt eine neue Seite anforderten, war kaum mehr CPU-Last bemerkbar, wobei der Rechner, auf dem die Browser liefen, deutlich mehr CPU-Leistung für die Darstellung brauchte.

### 6.3.4 Test auf Memory Leaks

Da es in C++ schnell zu Memory Leaks - auch Speicherfressern genannt - kommen kann, muss nach diesen gesucht werden. Dafür wurde jede Komponente mit Rational Purify for Windows getestet. Dieses Programm merkt sich jeden allokierten Speicherbereich, und beim Beenden des Programms wird der allokierte Speicher angezeigt.

# 7 Status und Ausblick

Da es für die existierenden Anforderungen des Slow Control Systems im Pierre Auger Observatorium keinen passenden Gateway gab, der die OPC DA Schnittstelle in die OPC XML-DA Schnittstelle umsetzt, wurde im Rahmen dieser Arbeit der so genannte „OPC XML-DA Gateway“ entworfen und umgesetzt.

Um auf diesen Gateway oder einer anderen OPC XML-DA Schnittstelle zugreifen zu können, wurde der so genannte OPC XML-DA Client entwickelt. Dieser Client ist so entwickelt worden, dass er unter den Betriebssystemen Linux und Windows einsetzbar ist. Auf die richtige Umsetzung der OPC XML-DA Spezifikation wurde der Client gegen einen OPC XML-DA Server der OPC Foundation getestet.

Für diesen Client gibt es noch eine Erweiterung in Form einer ausführbaren Datei, die die Variablen über eine „Shared Memory Schnittstelle“ bereitstellt und entgegennimmt.

Zur Darstellung und Steuerung von OPC XML-DA Schnittstellen wurde das SCADA System „SCS-HTTP-Server“ entworfen und entwickelt. Mit diesem Server können Element-Daten einer oder mehrerer OPC XML-DA Schnittstellen in HTML-Seiten dargestellt werden, bzw. Variablen aus der HTML-Seite über den Server in die OPC XML-DA Schnittstelle geschrieben werden.

Diese drei entstandenen Komponenten wurden ausgiebig in der dafür aufgebauten Umgebung getestet.

Im nächsten Schritt sollen diese Komponenten im Pierre Auger Observatorium in Argentinien getestet und in Betrieb genommen werden. Dafür sollten für den SCS-HTTP-Server die noch erforderlichen HTML-Seiten, die für den sicheren Betrieb der Slow Control Systeme benötigt werden, umgesetzt werden.

Der Zugriff auf die bereitgestellte Schnittstelle für die DAQ-Anwendung muss in dieser implementiert werden, da dieser direkte Zugriff auf das Slow Control System das Bediehnpersonal im CDAS entlasten wird.

Die Daten der Slow Control Systeme können durch den Einsatz der OPC XML-DA Client Komponente zentral abgefragt und in einer relationalen Datenbank gespeichert werden. Dadurch hätte man eine zentrale Datenarchivierung der Slow Control Systeme.

Die drei entwickelten Komponenten können als kommerzielles Produkt bereitgestellt werden. Besonders der OPC XML-DA Gateway mit dem SCS-HTTP-Server ist ideal, um einfache Visualisierungen und Steuerungen für OPC DA umzusetzen.

Eine andere Möglichkeit wäre, aus den vorhandenen Komponenten einen SCS-HTTP-Server zu bilden oder den bestehenden Server so zu erweitern, dass man direkt auf die OPC DA Schnittstelle zuzugreifen kann. Dieser ist dann aber nur unter dem Betriebssystem Windows einsetzbar, da OPC DA nur mit DCOM betrieben werden kann.

# Abkürzungsverzeichnis/Glossar

| | |
|---|---|
| ASP | Active Server Pages, (aktive Serverseiten), Webserver-Erweiterungen von Microsoft |
| BEEP | Blocks Extensible Exchange Protocol |
| CDAS | Central Data Aquisition System (Kontrollzentrum des Pierre Auger Observatorium) |
| CERN | Conseil Européenne pour la Recherche Nucléaire (Europäische Organisation für Kernforschung) |
| CLSID | Classidentifier |
| CPU | Central Processing Unit (Hauptprozessor) |
| DAQ | Data Acquisition (Programm zur Datenerfassung) |
| DCOM | Distributed Component Object Model ist ein von Microsoft definiertes Protokoll, um Programmkomponenten über ein Netzwerk kommunizieren zu lassen. |
| DLL | Dynamic Link Library (Dynamische Verbindungsbibliothek) |
| EXE | EXEcutable (ausführbare Datei) |
| FD | Fluoreszenzdetektor |
| FTP | File Transfer Protocol, ein Netzwerkprotokoll zur Dateiübertragung |
| GUID | Globally Unique Identifier ist eine global eindeutige Zahl, die in verteilten Computersystemen zum Einsatz kommt. |
| HTML | Die Hypertext Markup Language ist ein Dokumentenformat zur Auszeichnung von Hypertext im World Wide Web. |
| HTTP | Hypertext Transfer Protocol ist ein zustandsloses Protokoll zur Übertragung von Daten. |
| HV | High Voltage (Hochspannung) |
| IEC | International Electrotechnical Commission (Internationales Normierungsgremium) |
| IP | Internet Protocol (auch Internetprotokoll) |
| MB | Megabyte, eine Informationsmenge in der Digitaltechnik und Informatik |
| OLE | Object Linking and Embedding ist ein Standard, der von Microsoft entwickelt wurde und einen einfachen Datenaustausch zwischen verschiedenen OLE-fähigen Applikationen via Drag & Drop oder Copy & Paste ermöglicht. |
| OPC | OLE for Process Control, standardisierte Software-Schnittstelle |
| OPC AE | OPC Alarm and Event |
| OPC DA | OPC Data Access |
| OPC HDA | OPC Historical Data Access |

| | |
|---|---|
| OPC XML-DA | OPC Data Access mit SOAP |
| PC | Personal Computer |
| PROFIBUS | Process Field Bus ist ein Standard für die Feldbus-Kommunikation in der Automatisierungstechnik. |
| RAM | Random Access Memory (Arbeitsspeicher eines Computers) |
| RPC | Remote Procedure Call ist ein Netzwerkprotokoll. |
| SC | Slow Control System |
| SCADA | Supervisory Control and Data Acquisition (Steuerung und Visualisierung) |
| SCS | Slow Control SCADA |
| SD | Surface Detector (Oberflächenmessstation) |
| SMTP | Simple Mail Transfer Protocol ist ein Protokoll der TCP/IP-Protokollfamilie. |
| SOAP | Eigenname, ursprünglich die Abkürzung für Simple Object Access Protocol (Einfaches Objekt-Zugriffs-Protokoll) |
| SPS | Eine Speicherprogrammierbare Steuerung ist eine einem Computer ähnliche elektronische Baugruppe, die für Steuerungs- und Regelungsaufgaben in der Automatisierungstechnik eingesetzt wird. |
| TCP | Transmission Control Protocol |
| URI | Uniform Resource Identifier |
| USV | Unterbrechungsfreie Stromversorgung (USV) soll beim Ausfall der Netzspannung die Stromversorgung sicherstellen. |
| VNC | Virtual Network Computing (Fernwartungssoftware) |
| WSDL | Web Services Description Language definiert einen plattform-, programmiersprachen- und protokollunabhängigen XML-Standard zur Beschreibung von Netzwerkdiensten zum Austausch von Nachrichten. |
| XML | Extensible Markup Language ist ein Standard zur Erstellung maschinen- und menschenlesbarer Dokumente in Form einer Baumstruktur. |

# Literaturverzeichnis

[ALKHO75] Alkhofer, O. C.: Introduction to Cosmic Radiation, Carl Thiemig, München 1975.

[AUGER38] o. V.: Comtes rendus, Académie des Sciences 206, o. O. 1938, S. 1721.

[AUGER97] Auger Collaboration: The pierre auger project design report, second edition, o. O. 1997.

[AXIS] http://ws.apache.org/axis/cpp/documentation.html, letzter Zugriff: 20.8.2005.

[BARKH] Barkhausen, Martin, CD dieser Diplomarbeit.

[BECK05] http://www.beckhoff.de/german/busterm/aufbau.htm, letzter Zugriff: 20.8.2005.

[BHM+04] Booth D. et al.: Web Services Architektur, http://www.w3.org/TR/ws-arch/.

[BIRD95] Bird, D. J. et al.: Fly's Eye Kollaboration, Astrophys. J. 441, o. O. 1995.

[CAEN] http://www.caen.it/nuclear/product_list.php?cat=ps, letzter Zugriff: 20.8.2005.

[CER02] Cerami, E.: Web Services Essential: Distributed Application with XML-PRC, SOAP, UDDI and WSDL, O'Reilly, o. O. 2002.

[DIJKS72] Dijkstra, Edsger W.: Structured Programming, Academic Press, London / New York 1972.

[DOLL90] Doll, P. et al.: KASCADE Kollaboration, The Karlsruhe Cosmic Ray Projcet KASCADE, KfK-Report 4686, Karlsruhe 1990.

[FIELD05] http://softing.com/en/controls/products/fieldpc.htm, letzter Zugriff: 20.8.2005.

[GSOAP] http://www.cs.fsu.edu/~engelen/soap.html, letzter Zugriff: 20.8.2005.

[HESS12] Hess, V.: Physikalische Zeitschrift 13, o. O. 1912, S. 1084.

[IWANI00] Iwanitz, Frank u. Lange, Jürgen: OLE for Process Control: Grundlagen, Implementierung und Anwendung, Hüthig GmbH, o. O. 2000.

[KASSL] http://www.kassl.de/dopc/index.html, letzter Zugriff: 20.8.2005.

[OPCAE02] OPC Foundation, Alarm and Event Custom Interface Standard Version 1.1 (2002).

[OPCDA03] OPC Foundation, Data Access Custom Interface Standard Version 3.0 (2003).

[OPCHD03] OPC Foundation, OPC Historical Data Access Specification Version 1.2 (2003).

[OPCXM04] OPC Foundation, OPC XML-DA Specifikation Version 1.01 (2004).

[RFC2616] Network Working Group: RFC 2616 - Hypertext Transfer Protocol - HTTP/1.1, o. O. 1999.

[RFC821] Postel, Jonathan B.: RFC 821 - Simple Mail Transfer Protocol, o. O. 1982.

[RFC959] Network Working Group: RFC 959 - File Transfer Protocol, o. O 1985.

[SOAP03] W3C Consortium: SOAP 1.2 Spezifikation , o. O. 2003.

[SOFTI] http://www.softing.de, letzter Zugriff: 20.8.2005.

[TAKED03] Takeda, M. et al.: Astropart. Phys. 19, o. O. 2003, S. 447-462.

[TECHN] http://www.tswinc.us/pc-6-5-xml-da-server-side-gateway.aspx, letzter Zugriff: 20.8.2005.

[URI94] Lee, B. et al.: Universal Resource Identifiers (URI): Generic Syntax, RFC 2396, August 1994.

[VORGE] Fuchs, Gerhard: State of the Art - Seminare „Prozesse und Profile“, www3.informatik.uni-erlangen.de/ Lehre/UML-Seminar/SS2002/StateOfTheArt.ppt, letzter Zugriff: 20.8.2005, S. 16.

[WSDL01] W3C Consortium: Web Services Description Language (WSDL) 1.1, o. O. 2001.

[QT] http://www.trolltech.com/products/qt/index.html, letzter Zugriff: 20.8.2005.

[XERCE] http://xml.apache.org/xerces-c, letzter Zugriff: 20.8.2005.

[XML] http://www.w3.org/XML, letzter Zugriff: 20.8.2005.

# Inhalt der CD

Auf der beiliegenden CD befinden sich die Dateien zu dieser Arbeit. Der genaue Inhalt der CD wird in der Datei „readme.txt“ beschrieben.

# Eidesstattliche Erklärung

Martin Barkhausen, 872021

Hiermit erkläre ich, dass ich diese Arbeit selbständig abgefasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

_____ _____

Ort, Datum Unterschrift