CERN-THESIS-2008-187

Fachbereich Mathematik und Naturwissenschaften
Fachgruppe Physik
Bergische Universität Wuppertal

Entwurf und Implementation eines Expertensystems für das Detektorkontrollsystem des ATLAS-Pixeldetektors

Dissertation zur Erlangung des Doktorgrades
vorgelegt von
Tobias Henß

Dezember 2008

WUB-DIS 2008-08

Inhaltsverzeichnis

Einleitung

Das Gebiet der Hochenergiephysik beschäftigt sich mit den Elementarteilchen und den zwischen ihnen wirkenden Kräften. Dazu werden theoretische Modelle entwickelt und deren Aussagen mit Hilfe von Experimenten überprüft. Das Standardmodell der Teilchenphysik (siehe Kapitel 1) liefert seit den 80er Jahren des vergangenen Jahrhunderts präzise Vorhersagen über den Aufbau der Materie und ihre Wechselwirkungen, die bisher in zahlreichen Experimenten bestätigt werden konnten.

Je schwerer ein Elementarteilchen ist, umso höher muss dabei die von Teilchenbeschleunigern bereitgestellte Schwerpunktsenergie sein, um seine Eigenschaften und Existenz im Labor nachzuweisen. Das letzte vom Standardmodell vorhergesagte und bisher nicht entdeckte Elementarteilchen ist das Higgs-Boson, von dem angenommen wird, dass es für die Erzeugung der Teilchenmassen verantwortlich ist.

Mitte 2009 wird mit dem Large Hadron Collider am europäischen Teilchenphysik-Zentrum CERN[1] der bisher größte und leistungsfähigste Teilchenbeschleuniger, der LHC[2], in Betrieb genommen. An vier Wechselwirkungspunkten werden dort Protonen mit einer Schwerpunktsenergie von 14 TeV zur Kollision gebracht. Der ATLAS[3]-Detektor ist einer von vier großen Detektoren am LHC, mit deren Hilfe Physiker bisher ungelöste Probleme der Elementarteilchenphysik erforschen werden (siehe Kapitel 2).

Eine besondere Bedeutung bei der Rekonstruktion der genauen Kollisionsorte und der Impulse geladener Teilchen kommt dabei dem Pixeldetektor zu (Kapitel 3). Als innerster Teildetektor des ATLAS-Experiments ergeben sich für den Pixeldetektor besondere mechanische und elektrische Anforderungen, die eine komplexe Infrastruktur zum Betrieb des Detektors erfordern. Um die vielen tausend resultierenden Betriebsparameter, wie Spannungen, Ströme und Temperaturen, zu kontrollieren, wurde ein Detektorkontrollsystem DCS[4] entwickelt (Kapitel 4). Dieses besteht aus Hard- und Softwarekomponenten und ermöglicht die Kontrolle des Pixeldetektors durch eine einzelne Person.

[1] Aus der ursprünglichen französischen Bezeichnung: **C**onseil **E**uropéen de la **R**echerche **N**ucléaire
[2] **L**arge **H**adron **C**ollider
[3] Ursprünglich: **A** **T**oroidal **L**HC **A**paratu**S**. Inzwischen als Eigenname gebraucht.
[4] **D**etector **C**ontrol **S**ystem

Um möglichst viele Daten zu sammeln, werden die LHC-Experimente, und damit auch der Pixeldetektor, mehrere Jahre, von Wartungsphasen abgesehen, kontinuierlich, Tag und Nacht, betrieben. Der dadurch bedingte, langfristige Schichtbetrieb erfordert den Einsatz einer Vielzahl an Nicht-DCS-Experten in den Reihen des Betriebspersonals.

Im Rahmen dieser Arbeit wurde mit dem "Pixel-Advisor" daher ein Expertensystem (eine generelle Beschreibung des Begriffs bietet Kapitel 5) entworfen und implementiert, das es dem normalen DCS-Schichtpersonal durch die Bereitstellung regelbasierten Expertenwissens ermöglichen soll, Kontrollsystem-bezogene Probleme ohne die Hilfe eines DCS-Experten zu lösen.

Die Entwicklung eines solchen Expertensystems beinhaltete neben der genauen Aufstellung der zu erfüllenden Anforderungen (Kapitel 6), insbesondere auch umfangreiche Studien zur Tauglichkeit der zu verwendenden Programmierwerkzeuge und Hilfsmittel (Kapitel 7).

Dies ermöglichte den endgültigen Entwurf des Pixel-Advisors und seiner Komponenten, deren Implementation und Eigenschaften in der vorliegenden Dissertation beschrieben werden sollen (Kapitel 8).

Abschließend sollen die aus der Nutzung am CERN gewonnenen Erfahrungen, sowie dadurch motivierte Verbesserungen dargestellt und anhand einiger Beispiele ein Ausblick auf mögliche, zukünftige Weiterentwicklungen des Expertensystem gegeben werden (Kapitel 9).

Kapitel 1

Physik des Standardmodells

"Nur scheinbar hat ein Ding eine Farbe, nur scheinbar ist es süß oder bitter; in Wirklichkeit gibt es nur Atome und leeren Raum." [Demokrit]

1.1 Historie der Teilchenphysik

Bereits um 400 vor Christus unserer Zeitrechnung spekulierten griechische Philosophen, die sie umgebende Materie sei aus "Atomen" (von "Atomos" griech.: unteilbar) aufgebaut (Demokrit). Dennoch dauerte es mehr als 2000 Jahre, bis dieser Gedankengang erfolgreich aufgegriffen wurde.

- Das erste nach unserem heutigen Kenntnisstand unteilbare, also elementare Teilchen, das Elektron (von „Elektron" griech.: Bernstein), wurde 1897 von Joseph John Thomson durch Ablenkversuche an Kathodenstrahlen gefunden.

- Acht Jahre später formulierte Albert Einstein mit seiner Lichtquantenhypothese eine Erklärung für den photoelektrischen Effekt und führte damit sogleich das Photon (von „Phos" griech.: Licht) ein.

- 1909 machten Hans Geiger und Ernest Marsden bei der Streuung von Alpha-Teilchen an einer Goldfolie eine Entdeckung, die Ernest Rutherford 1911 zur Veröffentlichung seines, einen Atomkern beinhaltenden, Atommodells veranlasste. Er widerlegte damit die damals gängige Auffassung einer homogenen Massedichte innerhalb des Atoms.

- 1919 entdeckte Rutherford das Proton (von „Protos" griech.: der Erste) bei der Bestrahlung von Stickstoff mit Alphateilchen (es entstehen Wasserstoff- und Sauerstoffionen).

- Der atomare Gegenpart des Protons, das Neutron wurde 1932 von James Chadwick gefunden (er bestrahlte Beryllium mit Alpha-Teilchen und untersuchte die dabei entstehende Strahlung, die ursprünglich fälschlicherweise als Gamma-Strahlung interpretiert worden war).
- Im selben Jahr fand Carl David Anderson das Positron in der kosmischen Strahlung und erbrachte damit den Beweis für die bereits 1928 von Paul A. M. Dirac postulierte Existenz der Antimaterie.

Die Entwicklung neuer Technologien wie beispielsweise der Blasenkammer oder des Teilchenbeschleunigers führten in den darauf folgenden Jahrzehnten zu einer wahren Flut neu entdeckter Teilchen, deren Existenz vorerst im Widerspruch zur Theorie, eines auf wenigen elementaren Grundbausteinen aufbauenden Basismodells der Teilchenphysik stand.

Ein entscheidender Fortschritt in der Erklärung der Eigenschaften dieses "Teilchenzoos" gelang mit dem "Standardmodell" der Teilchenphysik, auf das im folgenden Abschnitt 1.2 näher eingegangen wird. Die Tatsache, dass viele der vom Standardmodell vorhergesagten physikalischen Eigenschaften mit einer hohen Präzision durch Experimente verifiziert werden konnten, trägt wesentlich zu unserem heutigen Vertrauen in das Standardmodell bei.

Neue, im Bau befindliche Experimente (siehe Kapitel 2) sollen dazu beitragen, die Erklärungen des Standardmodells für verschiedene offene Probleme, wie beispielsweise die Entstehung der Masse der Fermionen, entweder zu bestätigen oder zu Gunsten weiter gehender Theorien, wie etwa der Supersymmetrie zu erweitern oder zu verwerfen.

Bemerkung zu den verwendeten Einheiten

In der Hochenergiephysik werden in vielen Fällen keine SI[a]-Einheiten verwendet. Stattdessen kommen dem Problemfeld angepasste Einheiten zum Einsatz[14].
Drehimpulse werden üblicherweise in Bruchteilen des Planckschen Wirkungsquantums $\hbar$ angegeben:

$$\hbar = 1.054571628(53) \cdot 10^{-34}\ \text{Js}$$

Da zur Beschleunigung von Teilchen elektrische Felder eingesetzt werden, hat es sich in der Hochenergiephysik eingebürgert, Energien durch die zu deren Erreichen notwendige Beschleunigungsspannung anzugeben. Ein eV bezeichnet dabei die Energie eines Elektrons nach Durchlaufen einer Beschleunigungsspannung von 1 V.

$$1\ \text{eV} = 1.6022 \cdot 10^{-19}\ \text{J}$$

Gemäß der Einstein'schen Beziehung $E = m \cdot c^2$ sind Masse m und Energie E durch die Proportionalitätskonstante Lichtgeschwindigkeit c verbunden.

$$c = 299792458\ \text{m/s}$$

Man kann daher die häufig verwendeten Größen Masse und Impuls in Einheiten von eV und Bruchteilen von c angeben:

$$[m] = [\frac{E}{c^2}] = \frac{\text{eV}}{c^2}$$

$$[p] = [\frac{E \cdot v}{c^2}] = [\frac{E}{c}] = \frac{\text{eV}}{c}$$

Oftmals verwendet man überdies die Konvention $\hbar = c = 1$, so dass Energien, Massen und Impulse in Einheiten von eV angegeben werden.

[a] **S**ystème **I**nternational d'Unités, Internationales Einheitensystem

1.2 Das Standardmodell der Teilchenphysik

Das Standardmodell der Teilchenphysik beschreibt den Mikrokosmos durch einen Satz elementarer, also nicht weiter zerlegbarer Teilchen, sowie deren Wechselwirkungen. Die Welt, die uns umgibt besteht dabei aus Teilchen mit halbzahligem Eigendrehimpuls, auch „Spin“ genannt, den „Fermionen“[1], und Teilchen mit ganzzahligem Spin, den „Bosonen“[2].

[1] Zu Ehren von Enrico Fermi, da diese Teilchen der Fermi-Dirac-Statistik gehorchen.
[2] Zu Ehren von Jagadish Chandra Bose; Bosonen gehorchen der Bose-Einstein-Statistik.

1.2.1 Fermionen

Die Fermionen sind die Materieteilchen des Standardmodells und damit die Grundbausteine aller Materiestrukturen[3]. Neben der bereits erwähnten Eigenschaft des halbzahligen Spins besitzen die Fermionen auf Grund des Pauli-Prinzips die wichtige Eigenschaft, dass sich zwei Fermionen immer in mindestens einer ihrer Quantenzahlen unterscheiden müssen, sofern Ihre Wellenfunktionen überlappen. Dies führt beispielsweise zu den Besetzungszahlen atomaren Hüllenelektronen. Alle bekannte Materie ist aus fermionischen Grundbausteinen, den "Quarks" und "Leptonen" aufgebaut, die im Folgenden näher erläutert werden sollen.

1.2.1.1 Die vier Grundbausteine des Universums

Obwohl heute zwölf als elementar angesehene Fermionen bekannt sind, so würden doch vier fundamentale Fermionen ausreichen, die uns umgebende Materie aufzubauen. Die Frage, warum es darüber hinaus weitere, für uns nur unter Laborbedingungen sichtbare Fermionen gibt, ist eine der offenen Fragen der Teilchenphysik.

Das bekannteste dieser vier Teilchen ist das Elektron, dessen negative elektrische Ladung in der Atomhülle die positive Ladung der Protonen kompensiert.

Aus besagter Protonladung folgt, dass die Konstituenten des Protons selbst geladen sein müssen. Sowohl Proton, als auch Neutron lassen sich aus lediglich zwei verschiedenen Quarktypen aufbauen. Das "Up"-Quark trägt dabei die Ladung $+2/3$ und das "Down"-Quark die Ladung $-1/3$. Ein Proton besteht im einfachsten und in der Natur auch realisierten Fall aus der Quarkkombination uud und ein Neutron aus udd.

Die Erklärung des Betazerfalls, der auf der Quarkebene als Umwandlung eines Down-Quarks in ein Up-Quark geschrieben werden kann, verlangt aus Gründen der Ladungserhaltung die Abstrahlung eines Elektrons. Bereits 1911 konnten Lise Meitner und Otto Hahn zeigen, dass das beim Betazerfall abgestrahlte Elektron nicht wie erwartet, die durch den Zerfallsprozess vorgegebene, feste Energie besaß, sondern über ein kontinuierliches Spektrum verfügte. Die Erklärung dafür ist die Abstrahlung eines weiteren, lediglich schwach (siehe 1.2.2.2) wechselwirkenden Teilchens, des Neutrinos.

1.2.1.2 Die drei Fermionen-Familien

Die bisher eingeführten Fermionen bilden eine "Familie" aus zwei "Quarks" u, d und zwei "Leptonen" ν_e und e^-. Wie in Abschnitt 1.2.2.3 erklärt wird, handelt es sich bei den Quarks um die Fermionen, die an der starken Wechselwirkung teilnehmen, während die

[3] allerdings können aus Konstrukten mehrerer Fermionen durchaus bosonische Endzustände entstehen, siehe „Mesonen“ in Abschnitt 1.2.3

Leptonen dies nicht tun. Obwohl nur diese vier Teilchen stabil sind und damit am Aufbau der uns umgebenden Materie beteiligt sind, hat man noch zwei weitere Fermionen-Familien oder "Generationen" gefunden, die ebenfalls aus je vier Fermionen bestehen.

$$\begin{pmatrix} u \\ d \end{pmatrix} \begin{pmatrix} c \\ s \end{pmatrix} \begin{pmatrix} t \\ b \end{pmatrix}$$

$$\begin{pmatrix} \nu_e \\ e^- \end{pmatrix} \begin{pmatrix} \nu_\mu \\ \mu^- \end{pmatrix} \begin{pmatrix} \nu_\tau \\ \tau^- \end{pmatrix}$$

Bei den Teilchen der zweiten und dritten Familie handelt es sich dabei gewissermaßen um schwerere Kopien von u, d, ν_e und e^-. Das Massenspektrum der Teilchen der drei Familien reicht dabei von dem als nahezu masselos angesehenen Elektron-Neutrino ν_e bzw. dem mit einer Masse von 511 keV ebenfalls sehr leichten Elektron e bis hin zum t-Quark der dritten Generation, das mit einer Masse von mehr als 170 GeV so schwer ist wie ein Goldatom.

1.2.1.3 Antiteilchen

Jedes Fermion besitzt ein Antiteilchen, dass sich durch die entgegengesetzten Quantenzahlen von dem ursprünglichen Teilchen unterscheidet. Die Antiteilchen zu den in Tabelle 1.2 aufgeführten Fermionen erhält man demnach durch Multiplikation ihrer Ladungs- und Spinquantenzahlen mit -1.

1.2.2 Bosonen und Kräfte

Während die Fermionen die bekannte Materie bilden, dient eine zweite Gruppe von Elementarteilchen, die Bosonen, als Überträger der Wechselwirkungskräfte. Jeder fundamentalen Kraft sind dabei ein oder mehrere "Austauschbosonen" zugeordnet, deren Eigenschaften Reichweite und Kraft der zugehörigen Wechselwirkung bestimmen. Die Reichweite einer Kraft kann dabei durch die Form ihres zugehörigen Yukawa-Potentials[4] gekennzeichnet werden[5]:

$$V(r) = const \cdot \frac{e^{-mr}}{r} \tag{1.1}$$

[4]Nach dem japanischen Physiker Hideki Yukawa

[5]Im Falle der starken Kraft ist dieses Potential lediglich ein effektives Potential zur Beschreibung der Kernkraft durch den Austausch von Pionen. Für kleine Abstände und Gluonen als Austauschteilchen gilt dieses Potential nicht. Ebenso erhält man so auch für die schwache Kraft lediglich eine Abschätzung der Bosonenmassen.

Dabei ist m die Masse des beteiligten Austauschbosons, so dass das Potential für hohe Bosonmassen schneller abfällt und sich damit die effektive Reichweite verringert (siehe Tabelle 1.1). Man spricht in diesem Zusammenhang von einer begrenzten Reichweite des Potentials, sofern die beteiligten Austauschbosonen massebehaftet sind.

Das Standardmodell kennt 4 fundamentale Kräfte, die elektromagnetische, die schwache und die starke Kraft, sowie die Gravitation. Letztere spielt in der Teilchenphysik, auf Grund ihrer relativen Schwäche, jedoch nur eine sehr untergeordnete Rolle, weshalb die Gravitation im Allgemeinen vernachlässigt wird (siehe Tabelle 1.1).

Die in der Tabelle gegebenen Reichweiten der Fundamentalkräfte stimmen mit der, sich aus dem Yukawa-Potential ergebenden Abschätzung, bis auf den Fall der starken Kraft überein (offensichtlich sollte ein masseloses Boson eine Kraft unendlicher Reichweite übertragen). Wie in Abschnitt 1.2.2.3 beschrieben wird, gilt für die starke Kraft eine andere Argumentation, die hier noch nicht näher erläutert werden soll. Im Falle der elektromagnetischen Kraft führt die Masselosigkeit des Austauschbosons zum bekannten Coulombpotential.

Kraft	Starke	Elektromagnetische	Schwache	Gravitation
rel. Stärke	1	10^{-2}	10^{-13}	10^{-38}
Reichweite [m]	$2.5 * 10^{-15}$	∞	10^{-18}	∞
Bosonen	8 Gluonen	Photon	W^+, W^-, Z	Graviton
Bosonmasse [GeV/c^2]	0	0	80, 80, 91	0

Tabelle 1.1: *Die vier fundamentalen Kräfte*

Jede der erwähnten Kräfte wirkt nur auf solche Teilchen, die die entsprechende Kopplungseigenschaft besitzen. In Anlehnung an den elektromagnetischen Fall spricht man hier allgemein von starker, schwacher oder eben elektromagnetischer Ladung. Teilchen, die die jeweilige Ladung nicht tragen, nehmen an der entsprechenden Wechselwirkung nicht teil, man sagt auch, die Eichbosonen der zugehörigen Kraft „koppeln“ nicht an diese Teilchen (siehe Tabelle 1.2).

1.2.2.1 Die elektromagnetische Kraft und das Photon

Die elektromagnetische Wechselwirkung ist von den drei verbleibenden Kräften diejenige, die sich am leichtesten im Alltag erfahren lässt, ist sie doch für die Mehrzahl aller Phänomene der Alltagsphysik verantwortlich. So sind beispielsweise die Elektronen der Atomhülle über die elektromagnetische Kraft mit den Protonen des Atomkerns verbunden. Da alle Fermionen mit Ausnahme der Neutrinos über eine elektromagnetische Ladung verfügen, wechselwirken sie alle elektromagnetisch.

Das Austauschteilchen der elektromagnetischen Kraft ist das Photon, dessen Masselosigkeit zur unendlichen Reichweite der elektromagnetischen Kraft führt.

Teilchen	Spin [$\hbar$]	el. Ladung [e]	Masse [MeV/c^2]	starke WW	elektromagnetische WW	schwache WW
u	1/2	2/3	~ 3	ja	ja	ja
d	1/2	-1/3	~ 6	ja	ja	ja
c	1/2	2/3	~ 1200	ja	ja	ja
s	1/2	-1/3	~ 100	ja	ja	ja
t	1/2	2/3	~ 178000	ja	ja	ja
b	1/2	-1/3	~ 4300	ja	ja	ja
ν_e	1/2	0	$\leq 3 \cdot 10^{-6}$	nein	nein	ja
e^-	1/2	-1	0.511	nein	ja	ja
ν_μ	1/2	0	≤ 0.19	nein	nein	ja
μ^-	1/2	-1	105.66	nein	ja	ja
ν_τ	1/2	0	≤ 18	nein	nein	ja
τ^-	1/2	-1	1777.0	nein	ja	ja

Tabelle 1.2: *Die Eigenschaften der elementaren Fermionen*

1.2.2.2 Die schwache Kraft und die Vektorbosonen

Die schwache Kraft ist verantwortlich für den radioaktiven Betazerfall der Atomkerne und trägt den Namen „schwach“ auf Grund der hohen Masse ihrer, auch Vektorbosonen genannten, Bosonen. Diese führt zu einer vergleichsweise kurzen Reichweite und einer Unterdrückung schwacher Prozesse gemäß der Heisenbergschen Unschärferelation[6] $\Delta E \cdot \Delta t \geq \bar{h}$.

Einzig über Prozesse der schwachen Kraft ist ein Änderung der Quark-“Sorte”, also beispielsweise eines u-Quarks in ein d-Quark möglich. Die beiden geladenen Austauschbosonen der schwachen Kraft W^+ und W^- koppeln nur an solche Quarkkombinationen deren der Ladungsunterschied gerade einer Elementarladung entspricht. Die geladenen W^+, W^- koppeln also jeweils nur an ein up- und ein down-artiges Quark. Im Gegensatz dazu erlaubt das neutrale Z die Kopplung an Quarks betragsmäßig gleicher Ladung. Da bisher keine Quarksorten-ändernden, neutralen Übergänge zwischen beispielsweise einem c und einem t Quark beobachtet wurden, kann diese Einschränkung sogar auf die Kopplung auf identische Quarks, bzw. Quark/Antiquark-Paare erweitert werden.

1.2.2.3 Die starke Kraft und die Gluonen

Die starke Kraft ist für die Bindung der Quarks zu Nukleonen und indirekt für die Bindung der Protonen und Neutronen zu Atomkernen verantwortlich. Die masselosen Gluonen, die Eichbosonen der starken Kraft, koppeln dabei an die “Farbladung” der Quarks. Die Ana-

[6] Um die gegebene Relation bei einer W-Masse von 80 GeV erfüllen zu können, ist eine maximale Wechselwirkungsdauer von lediglich etwa $8 \cdot 10^{-27}$ s erlaubt.

logie zur Farbe ergibt sich im Falle der starken Ladung daraus, dass es drei verschiedene Ladungszustände analog zu den drei Grundfarben des Farbspektrums gibt. Durch Quarks gebildete farbneutrale, also "weiße" Kombinationen aus diesen drei Farbladungen, etwa "rot", "grün" und "blau" oder auch "rot" und "anti-rot" nehmen hingegen nicht an der starken Wechselwirkung teil.

Da die Gluonen selbst eine Farbladung aus Farbe und Antifarbe tragen, koppeln sie sowohl an Quarks, als auch an andere Gluonen. Im Gegensatz zu allen anderen bekannten Kräften vermitteln die Gluonen ein für große Abstände (größer als 10^{-15} m) ansteigendes Kraftgesetz. Das führt dazu, dass die als Bindungsenergie im System gespeicherte Arbeit bei dem Versuch, zwei farbgeladene Teilchen zu trennen, mit steigendem Abstand kontinuierlich ansteigt. Bereits bei sehr kleinen Abständen reicht diese Energie aus, um zwischen den beiden Teilchen ein neues Teilchen-Antiteilchen-Paar zu erzeugen. Man sieht leicht ein, dass sich daher nur farbneutrale Kombinationen mehrerer Quarks frei bewegen können, was als Quark-"Confinement" bezeichnet wird.

Im Gegensatz zu den Quarks tragen die Leptonen keine Farbladung, nehmen also nicht an der starken Wechselwirkung teil.

1.2.3 Hadronisierung und Jets

Auf Grund des bereits erwähnten Confinements können Quarks nicht weiter als einige Femtometer voneinander entfernt werden, bevor sich entlang Ihrer Flugbahn neue farbgeladene Teilchen (Quarks oder Gluonen) bilden. Diesen Prozess bezeichnet man als Hadronisierung. Dabei bezeichnet man die farbneutralen und damit bezüglich der starken Wechselwirkung quasi-freien Quarksysteme als Hadronen. Da es neben den drei Farbladungen auch drei zugehörige "Antifarben" gibt, sind farbneutrale Kombinationen aus zwei oder mehr Quarks möglich.

Die aus Quark und Antiquark bestehenden Hadronen nennt man Mesonen, zu denen auch die in der kosmischen Höhenstrahlung vorkommenden Pionen, die sich aus Mischungen von Up- und Down-Quark zusammensetzen, zählen. Gebundene Zustände dreier Quarks werden als Baryonen bezeichnet, wobei die beiden bekanntesten das Proton und das Neutron sind. Theoretisch sind auch gebundene Zustände aus mehr als drei Quarks denkbar, diese wurden jedoch bisher experimentell nicht gefunden.

Die bei der Hadronisierung entstehenden Hadronen sind zwar farbneutral und daher nicht mehr über die starke Kraft an die ursprünglichen Quarks gebunden, haben jedoch keine gleichmäßige, den ganzen Raumwinkel abdeckende Winkelverteilung. Auf Grund der Energie- und Impulserhaltung sind sie in Flugrichtung des sich ursprünglich vom Wechselwirkungspunkt entfernenden Quarks konzentriert. Da bei der Hadronisierung eines vom Vertex ausgehenden Quarks sehr viele Hadronen erzeugt werden, deren Spuren innerhalb eines kleinen Raumwinkels liegen, spricht man auch von "Jets".

Einzelne Quarks können nicht direkt im Detektor nachgewiesen werden, da die Hadronisierung bereits wenige Femtometer vom Wechselwirkungsvertex entfernt beginnt. Die genaue Bestimmung der Eigenschaften eines Jets erlaubt es jedoch, gewisse Rückschlüsse auf die Art des ursprünglichen Quarks zu ziehen.

Eine Besonderheit bilden dabei die b-Quarks, die zunächst ein "B-Meson" (ein Meson, dass ein b-Quark enthält) bilden. B-Mesonen haben eine Lebensdauer von wenigen Pikosekunden und sind damit in der Lage, sich mehrere Millimeter vom primären Wechselwirkungspunkt zu entfernen, bevor sie in einem sekundären Vertex zerfallen. Der im Rahmen dieser Arbeit behandelte "Pixeldetektor" ist in der Lage, diese sekundären Vertizes vom Ort der primären Reaktion zu unterscheiden (siehe Kapitel 3).

1.2.4 Das Higgs-Boson

Das letzte bisher nicht gefundene Standardmodell-Teilchen ist das Higgs-Boson. Mit Hilfe des Higgs-Bosons lassen sich die bisher ansonsten nicht erklärbaren Teilchenmassen auf fundamentale Kopplungsterme zwischen dem Higgs und den einzelnen Fermionen und Bosonen zurückführen. Diese erlauben die Erklärung der Elementarteilchenmassen durch die Wechselwirkung mit einem Higgs-Feld.

Obwohl das Higgs-Teilchen bisher nicht entdeckt wurde, lässt sich der erlaubte Massenbereich für ein mit dem Standardmodell konformes Higgs-Boson eingrenzen. Die theoretischen Argumente für eine solche Einschränkung des Higgs-Massenbereichs folgen aus den Forderungen, dass eine störungstheoretische Behandlung möglich sein soll und dass das Standardmodell bis zu einer Energieskala[7] von $\Lambda_{GUT} \approx 10^{16}$ GeV seine Gültigkeit behält. Unter diesen Voraussetzungen lässt sich die Higgsmasse anhand theoretischer Argumente auf $m_H \leq 190$ GeV[14] eingrenzen. Direkte Messungen am LEP[8] sind in der Lage, Higgsmassen unterhalb von 114 GeV [27] auszuschließen, während eine Kombination indirekter Präzisionsexperimente eine obere Massengrenze von 185 GeV festlegt[23][39].

Diese aus Theorie und Experiment gewonnenen Massengrenzen motivieren den Bau neuer Experimente, mit denen die Teilchenphysiker in den genannten Energiebereich vordringen werden (siehe Kapitel 2). Mit einer geplanten Schwerpunktsenergie von 14 TeV ist der am CERN mit dem LHC geplante Proton-Proton-Beschleuniger in der Lage, Standardmodell-Higgs-Bosonen über den gesamten vorhergesagten Massenbereich zu produzieren.

Die vier für den LHC wichtigsten Produktionsprozesse des Standardmodell-Higgs-Bosons sind in Abbildung 1.1 aufgeführt. Der linke Plot in Abbildung 1.2 zeigt die Wirkungsquerschnitte dieser relevanten Produktionsprozesse. Die am LHC dominanten Pro-

[7] Diese Energieskala ist die Energie, bei der nach den "**G**rand **U**nified **T**heorie"-Vorhersagen die drei Kraftkopplungen in etwa gleich stark werden.

[8] **L**arge **E**lektron **P**ositron Collider

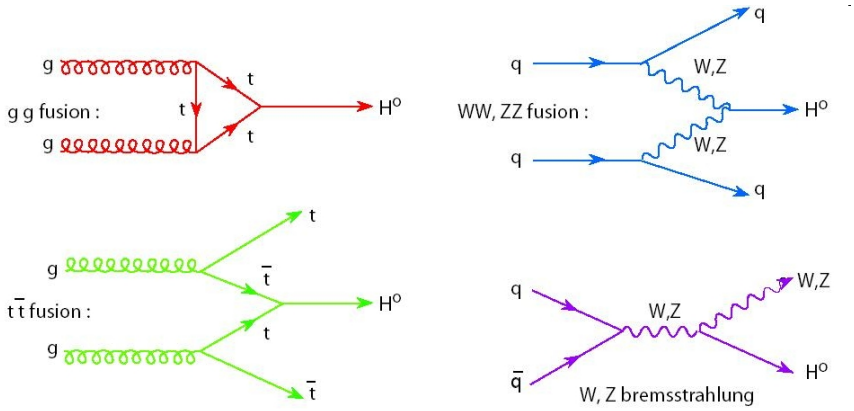


Abbildung 1.1: *Die vier wichtigsten Produktionsprozesse für ein Standardmodell-Higgs-Boson am LHC[35].*

duktionsprozesse sind dabei die Gluon- und Vektorboson-Fusion (auf den Abbildungen rot und blau eingezeichnet).

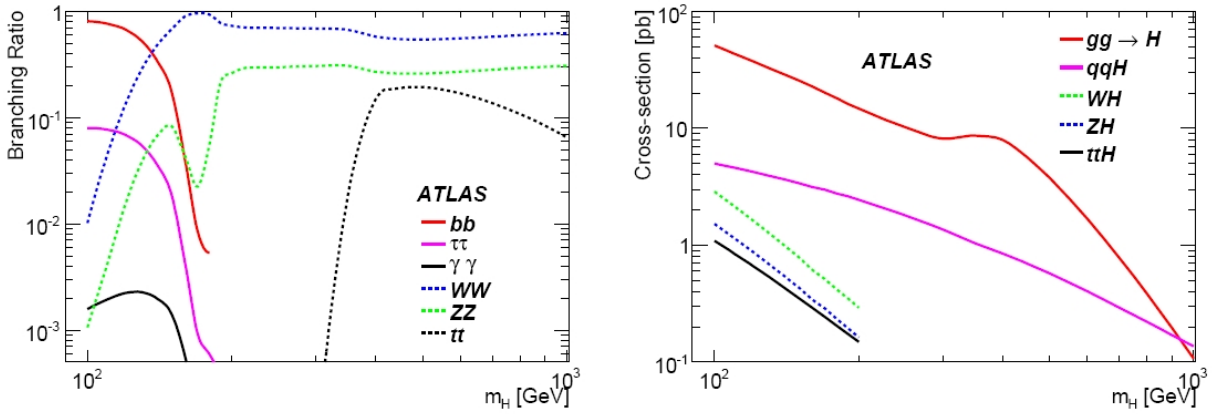


Abbildung 1.2: *Verzweigungsverhältnisse und Produktionskanäle des Higgs-Bosons als Funktion seiner Masse bei einer Schwerpunktsenergie von* 14 *TeV [9].*

Um die Masse des Higgs-Bosons zu ermitteln, muss diese aus der Messung seiner Zerfallsteilchen rekonstruiert werden. Ähnlich der Produktion existieren mehrere miteinander konkurrierende Zerfallskanäle, deren Verzweigungsverhältnisse in Abbildung 1.2 links aufgetragen sind. Insbesondere im unteren Bereich des erwarteten Massenintervalls ist dabei der Zerfall des Higgs in ein $b\bar{b}$-Paar dominant. Da am LHC QCD[9]-Untergrundprozesse eine hohe Rate an $b\bar{b}$-Paare erzeugen werden, ist es notwendig, nach einer aussagekräftigen Signatur im Endzustand des Prozesses zu suchen.

Ein wichtiger Kanal ist dabei durch den Prozess $t\bar{t}H \rightarrow t\bar{t}b\bar{b} \rightarrow 2 \cdot b\bar{b} + 2 \cdot W \rightarrow 2 \cdot b\bar{b} + q\bar{q} + l + \nu_l$ gegeben, dessen Feynman-Graph in Abbildung 1.3 gezeigt ist. Durch

[9] **Q**uantum **C**hromo **D**ynamics

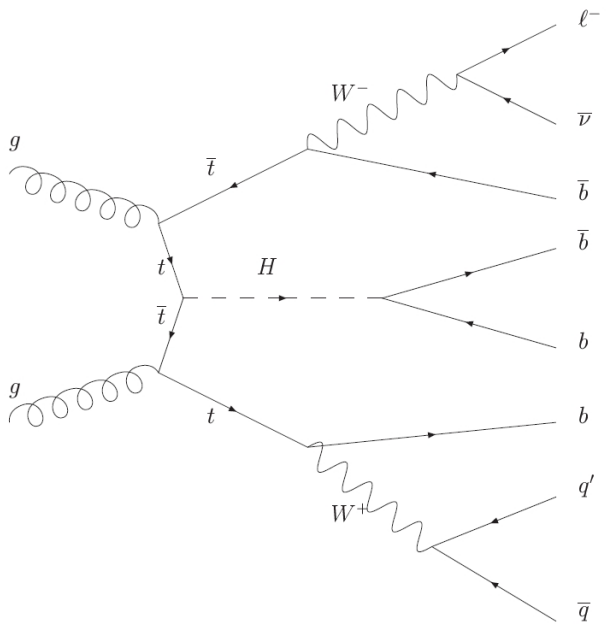


Abbildung 1.3: *Feynman-Graph des semileptonischen Zerfalls* $t\bar{t}H \rightarrow t\bar{t}b\bar{b} \rightarrow 2b\bar{b} + q\bar{q} + l + \nu$*[9].*

die vier b-Jets und das hochenergetische Lepton im Endzustand bietet dieser Kanal eine sehr gute Möglichkeit, den erwähnten Untergrund zu unterdrücken. Die wichtigste Aufgabe kommt dabei der korrekten Identifikation der b-Jets zu, da nur die Rekonstruktion aller vier b-Jets zu der gewünschten Untergrund-Reduktion führt[9]. Insbesondere die "b-Tagging"-Effizienz, also die Wahrscheinlichkeit einen b-Jet im Detektor auch als solchen zu rekonstruieren, spielt dabei eine entscheidende Rolle. Die Identifikation von b-Jets wird durch die Tatsache ermöglicht, dass die b-Quarks zunächst in B-Mesonen hadronisieren, die mit einer mittleren Lebensdauer von $(1.638 \pm 0.011) \cdot 10^{-12}$ s (B^+, B^-) bzw. $(1.530 \pm 0.009) \cdot 10^{-12}$ s (B^0)[14] langlebig genug sind, um erst nach einer Flugstrecke von:

$$s_{svtx} = v_B \cdot \tau'_B \approx c \cdot \gamma \cdot \tau_B \approx c\frac{E_B}{m_{0B}c^2}\tau_B \approx c\frac{30\ \text{GeV}}{5\ \text{GeV}}\tau_B \approx 3\ \text{mm} \tag{1.2}$$

in einem sekundären Vertex zu zerfallen. Mit Hilfe von "Vertexdetektoren", zu denen der im Rahmen dieser Arbeit vorgestellte ATLAS-Pixeldetektor gehört, lässt sich dieser sekundäre Zerfallsknoten von dem primären Proton-Proton-Wechselwirkungspunkt isolieren und erlaubt damit die Identifikation von b-Jets.

1.3 Moderne Experimente

Nach der, aus heutiger Sicht etwas unglücklich wirkenden, Entdeckung der Radioaktivität durch Henri Becquerel (er hatte zufällig einige radioaktive Präparate auf einer Fotoplatte abgelegt, die daraufhin eine Schwärzung aufwies) im Jahre 1896, wurden nach und nach immer ausgeklügeltere Verfahren zum Teilchennachweis entwickelt.

Die dabei noch heute genutzten Maschinen kann man in zwei Grundtypen, die Teilchenbeschleuniger und die Teilchendetektoren einteilen, deren Funktionsweise in den folgenden beiden Unterabschnitten 1.3.1 und 1.3.2 kurz erläutert werden soll.

Das Kapitel 2 wird dann konkrete Beispiele für beide Typen präsentieren.

1.3.1 Beschleuniger

Die moderne Teilchenphysik gliedert sich in zwei Hauptäste, die Astroteilchenphysik und die Hochenergiephysik (wobei letzterer Begriff streng genommen ungenau ist, da die höchsten bisher gemessenen Energien in Experimenten der Astroteilchenphysik nachgewiesen wurden). Während die Astroteilchenphysiker Detektoren für den Nachweis kosmischer Teilchen betreiben, erzeugen die Hochenergiephysiker die Rahmenbedingungen für ihre Experimente an den verschiedenen Beschleunigern selbst.

Ein Teilchenbeschleuniger beruht dabei immer auf dem Prinzip der Beschleunigung durch elektromagnetische Felder und setzt daher immer auf Beschleunigung geladener Teilchen.

Der naiv gesehen sicherlich einfachste Beschleunigertyp ist dabei der so genannte Linearbeschleuniger, bei dem die Teilchen auf einer geraden Flugstrecke beschleunigt werden. Der wesentliche Nachteil des Linearbeschleunigers liegt in der für die gewünschten Teilchenenergien benötigten enormen Länge der Beschleunigungsstrecke von etlichen Kilometern, was zu sehr hohen Baukosten führt. Zudem benötigt auch ein solcher, mit dem ILC[10] geplanter, ca. 40 Kilometer langer Linearbeschleuniger benötigt extrem hohe und daher technisch sehr anspruchsvolle Feldgradienten[11].

Auf diese extrem hohen Feldgradienten kann man beim zweiten Beschleunigertyp, dem Kreisbeschleuniger, weitgehend verzichten[12]. Hier durchlaufen die Teilchen bei jedem Umlauf immer wieder dieselben beschleunigenden Strukturen, so dass die Gesamtbeschleunigungsstrecke eines Kreisbeschleunigers meist weit länger ist (wenn man sie sich abgerollt

[10] **I**nternational **L**inear **C**ollider

[11] Der geplante ILC soll eine Schwerpunktsenergie von 500 GeV erreichen und benötigt dafür einen durchschnittlichen Feldgradienten von 31.5 MV/m über eine Beschleunigungslänge von 22 km[4].

[12] Dieses Argument gilt zumindest für die Beschleunigung von Hadronen. Im Falle leichter Leptonen können die hohen Synchrotronstrahlungsverluste (siehe folgender Absatz) ebenfalls den Einsatz hoher Feldgradienten erforderlich machen.

wie einen Faden vorstellt). Ein weiterer Vorteil der Kreisbeschleuniger liegt in ihrer Fähigkeit, die beschleunigten Teilchen für viele Umläufe bei relativ konstanter Energie zu speichern, weshalb diese Beschleunigerart auch "Speicherring" genannt wird.

Allerdings werden diese Vorteile durch beim Ablenken geladener Teilchen unvermeidlich entstehende "Synchrotronstrahlung" erkauft. Die pro Umlauf von den Teilchen emittierte Synchrotronstrahlung führt zu Energieverlusten, die durch die Beschleunigerstrukturen (zumeist Hochfrequenz-Kavitäten) ausgeglichen werden müssen. Die pro Umlauf und Teilchen emittierte Energie beträgt dabei[**?**]:

$$\Delta E = \frac{4\pi}{3}\frac{q^2\beta^2\gamma^4}{R} \tag{1.3}$$

Wobei q die Ladung, β die Teilchengeschwindigkeit in Bruchteilen der Lichtgeschwindigkeit und $\gamma = (1-\beta)^{-1/2}$ ist. Bei gleichem Impuls beträgt das Verhältnis des Energieverlustes zweier unterschiedlich schwerer, vom Betrag her gleich geladener Teilchen also:

$$\frac{\Delta E_m}{\Delta E_M} = \left(\frac{M}{m}\right)^4 \tag{1.4}$$

Für ein Elektron mit dem gleichen Impuls wie ein Proton ergibt sich also ein $(2\cdot 10^3)^4 = 1.6 * 10^{13}$ höherer Energieverlust.

Weiterhin werden die Teilchen durch starke Magnetfelder auf ihrer kreisförmigen Sollbahn gehalten. Bei gegebenem Bahnradius (weil beispielsweise der Beschleunigertunnel bereits existiert) und unter Vernachlässigung der Synchrotronstrahlung, hängt die maximal erreichbare Teilchenenergie mit der Stärke der ablenkenden Magnetfelder zusammen.

$$E_{max} = q(\vec{v} \times \vec{B}) \cdot \vec{R} \tag{1.5}$$

Im Falle beschleunigter Elektronen wird die erreichbare Teilchenenergie durch die in Formel 1.3 gegebenen Energieverluste begrenzt. Auf Grund ihrer höheren Masse spielt Synchrotronstrahlung bei Protonen eine untergeordnete Rolle, weshalb die höchsten bisher an Beschleunigern möglichen Energien mit Protonen erreicht werden. Deren Teilchenenergie wird dann im wesentlichen nur noch von der verfügbaren Magnetfeldstärke bestimmt.

Die frühe Version der Kreisbeschleuniger waren die "Zyklotrone", die ein gleichbleibend starkes Magnetfeld für die Ablenkung der Teilchen nutzten. Da der Radius der Teilchenflugbahn mit zunehmender Teilchenenergie in einem Zyklotron ansteigt, ist die maximal erreichbare Energie vor allem durch die Ausdehnung des Magnetfeldes begrenzt. Für die Beschleunigung niederenergetischer Teilchen braucht man sehr kleine Radien, während diese mit zunehmender Teilchenenergie vergrößert werden müssen. Daher müssen Zyklotrone in Scheibenform konstruiert werden, was für große Radien und damit Energien technisch

und wirtschaftlich aufwändig ist, weshalb diese heute nicht mehr in der Hochenergiephysik eingesetzt werden.

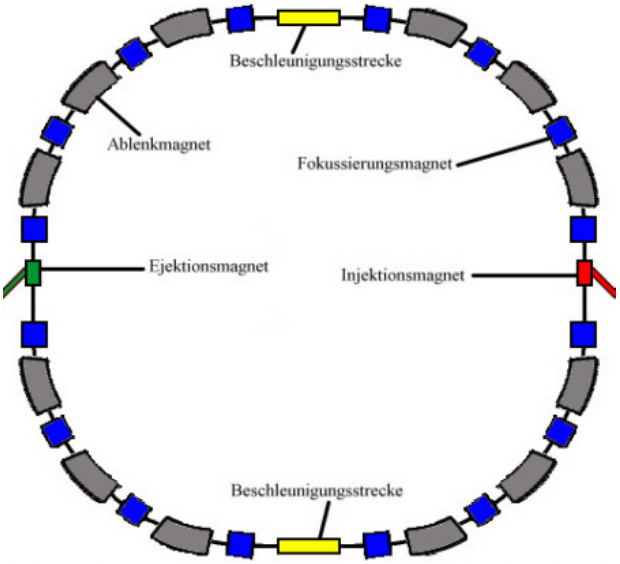


Abbildung 1.4: *Aufbauprinzip eines Synchrotron-Beschleunigers[25].*

Eine andere Möglichkeit der kreisförmigen Beschleunigung bietet das Synchrotron, bei dem der Radius der Kreisbahn im Gegensatz zum Zyklotron konstant gehalten wird. Um dennoch eine Beschleunigung über mehrere Umläufe zu erreichen, muss hierbei die Frequenz der Beschleunigungsspannung synchron (daher der Name) zur Umlauffrequenz der Teilchen erhöht werden. Des weiteren müssen die zur Ablenkung auf eine Kreisbahn benötigten Magnetfelder ebenfalls, gemäß Gleichung 1.5 synchronisiert werden. Auf Grund dieser Unterschiede benötigen Synchrotrone lediglich einen Beschleunigungsring mit konstantem Radius, was zur Senkung der Baukosten beiträgt. Die übliche Bauform ist dabei ein ringförmiger Tunnel der sowohl die Beschleunigungskavitäten, als auch die Ablenkmagneten und die Strahloptik enthält (siehe Abbildung 1.4).

1.3.2 Detektoren

Ähnlich der Situation bei den Beschleunigern, gibt es auch bei den zum Nachweis der Teilchen notwendigen Detektoren zwei konkurrierende Konzepte.

Bei “Fixed Target”-Experimenten werden die aus einem Beschleuniger extrahierten, hochenergetischen Teilchen auf ein festes Ziel, beispielsweise einen Wasserstofftank, gelenkt. Daraus resultiert eine sehr stark um die ursprüngliche Teilchenrichtung konzentrierte Winkelverteilung der Stoßpartner, was dazu führt, dass oftmals nicht der gesamte

Raumwinkel instrumentiert wird. Ausgehend von einer Strahlenergie E ergibt sich für die zur Erzeugung neuer Teilchen zur Verfügung stehende Schwerpunktsenergie:

$$\sqrt{s} \cong \sqrt{2E \cdot m_{target}} \tag{1.6}$$

Alternativ lässt man in einem „Collider“-Experiment zwei gegenläufige Teilchenstrahlen miteinander kollidieren und instrumentiert, idealerweise, den gesamten Raumwinkel um den “Primärvertex” genannten Kollisionspunkt. Der große Vorteil dieser Anordnung liegt in der höheren, erreichbaren Schwerpunktsenergie, die sich im Falle gleicher Energie der beiden Strahlen wie folgt berechnet:

$$\sqrt{s} \cong 2E \tag{1.7}$$

Der Nachweis der am Primärvertex abgelaufenen Reaktion lässt sich in den meisten Fällen nur sehr indirekt führen, da die primären Kollisionsteilchen auf Grund ihrer kurzen Lebensdauer (und damit Flugstrecke) entweder räumlich nicht aufgelöst werden können oder aber den Detektor (vom Vertex aus gesehen) gar nicht erst erreichen. Allerdings erlauben die Gesetze der Impuls-, Energie-, Ladungs- und Drehimpulserhaltung es, allein durch die genaue Vermessung von Energie, Impuls, Ladung und Winkelverteilung der sichtbaren Reaktionsprodukte, Rückschlüsse auf das primäre Ereignis und die Art der beteiligten Teilchen zu ziehen.

Ausgehend von dieser indirekten Vorgehensweise bestehen heutige Detektoren aus einer ganzen Anzahl hochspezialisierter „Subdetektoren“, die jeweils für ganz spezielle Unteraufgaben innerhalb der Rekonstruktion des Ereignisses zuständig sind. Abbildung 1.5 zeigt die Subdetektoren des in Kapitel 2 beschriebenen ATLAS-Experiments.

“Spur”- oder auch “Tracking”-Detektoren werden vornehmlich in der Nähe des Wechselwirkungspunktes eingesetzt. Ihre Aufgabe ist die genaue Vermessung der Flugbahnen geladener Teilchen. Durch den Einsatz starker ablenkender Magnetfelder und mit Hilfe komplexer Algorithmen lassen sich aus den Spurdaten weitere Informationen, wie beispielsweise der Impuls der Teilchen oder die Zugehörigkeit zu einem bestimmten Vertex (siehe “b-Tagging” im Abschnitt 1.2.4) ableiten.

“Kalorimeter” hingegen eignen sich primär zur Bestimmung der Energie von Teilchen, bzw. Jets. Ein Jet bezeichnet dabei ein sich in einem gewissen gemeinsamen Raumkegel ausbreitendes Teilchenbündel, dass auf einen gemeinsamen Vertex zurückgeführt werden kann (siehe Abschnitt 1.2.3). Auf Grund ihrer im Vergleich zu den Spurdetektoren meist recht groben räumlichen Granularität eignen sie sich weniger für die Positions- bzw. Impulsbestimmung. Elektromagnetische Kalorimeter können neben geladenen Teilchen auch Photonen registrieren. Elektronen und Photonen werden dabei vollständig gestoppt, bzw. absorbiert während andere, hinreichend energetische, ungeladene Teilchen das elektroma-

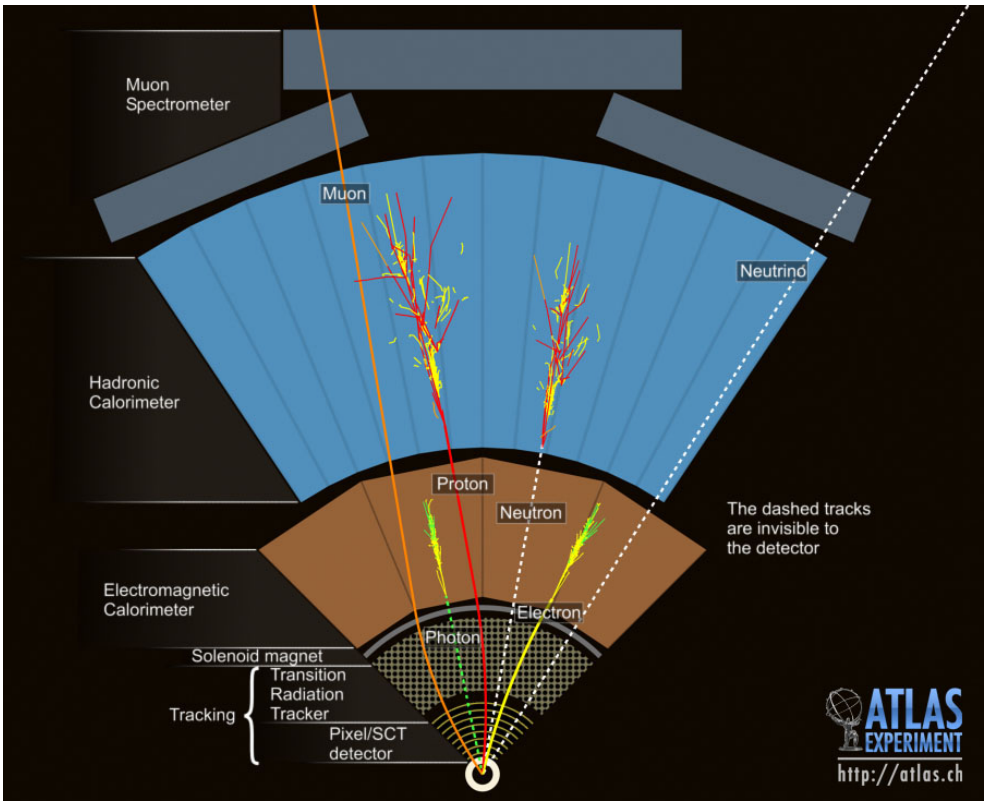


Abbildung 1.5: *Ausschnitt aus einem stark vereinfachten Modell des ATLAS-Detektors mit Darstellung eines Events[33].*

gnetische Kalorimeter durchdringen können. Diese werden im darauf folgenden hadronischen Kalorimeter registriert und gestoppt.

Um auch die Myonen, die alle bisher genannten Detektortypen passieren, zu registrieren, kommen außerhalb der Kalorimeter zusätzliche Spurdetektoren zum Einsatz, die im Bild als "Muon Spectrometer" bezeichnet werden. Auch im Volumen der Myonendetektoren kommen üblicherweise starke Magnetfelder zum Einsatz, um die Myonimpulse aus der Krümmung ihrer Flugbahnen rekonstruieren zu können. Konkrete Beispiele für die genannten Detektortypen werden in Kapitel 2 aufgeführt.

Funktionsprinzipien der Detektoren

Spurdetektoren arbeiten zumeist mit **Ionisationskammern**. Das Funktionsprinzip einer solchen Kammer beruht dabei immer auf der Erzeugung von freien Ladungsträgerpaaren durch ionisierende Teilchen in einem geeigneten Medium. Durch das Anlegen einer äußeren Spannung können diese Ladungsträger an den Rand der Ionisationskammer transportiert und dort mit Hilfe von Elektroden gesammelt werden. **Halbleiterdetektoren** und **Gaskammerdetektoren** unterscheiden sich dabei in der Art der Signalverstärkung, nicht aber in ihrem grundsätzlichen Messprinzip (siehe Abbildung 1.6).

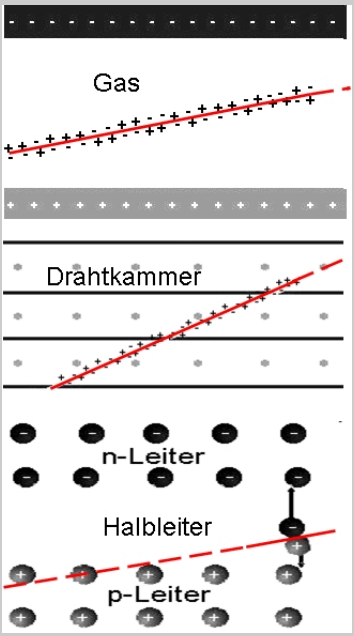


Abbildung 1.6: *Verschiedene Arten von Ionisationskammern[25].*

Die mit einem Spurdetektor erreichbare **Ortsauflösung** hängt von der **Segmentierung** des Detektors in einzelne Sensorzellen ab. Da sich mit Halbleiterdetektoren im Vergleich die kleinsten Sensorzellen realisieren lassen, besitzen diese die höchste Ortsauflösung.

Für die Spurdetektoren gilt dabei, dass durchfliegende Teilchen einem möglichst geringen Energieverlust unterliegen sollten, um beispielsweise die darauffolgende Energiemessung in den Kalorimetern nicht zu beeinträchtigen. Zur Charakterisierung dieser "Durchlässigkeit" werden die Begriffe **Strahlungslänge** und **Wechselwirkungslänge** genutzt. Die Strahlungslänge gibt dabei die Flugstrecke an, innerhalb derer die Energie eines Teilchens auf Grund elektromagnetischer Wechselwirkungen mit dem umgebenden Material auf $\frac{1}{e}$ seiner ursprünglichen Energie abgefallen ist. Die Wechselwirkungslänge ist analog definiert, bezieht sich allerdings auf den Energieverlust durch hadronische Wechselwirkungen.

Kalorimeter hingegen sind darauf angewiesen, die zu messenden Teilchen innerhalb ihres Detektorvolumens zu stoppen, um die gesamte Energie eines Teilchens zu registrieren. Kalorimeter nutzen die Bildung elektromagnetischer bzw. hadronischer "Schauer" aus, die entstehen, wenn hochenergetische Teilchen in dichte Materie eindringen. Dabei entstehen durch verschiedene Prozesse ganze Kaskaden von Folgeteilchen, bis die Energie sich soweit auf diese verteilt hat, dass diese im Material stoppen. Homogene Kalorimeter müssen, um auf eine ausreichende Zahl von Strahlungs- bzw. Wechselwirkungslängen zu kommen, entweder Sensormaterialien hoher Dichte (bspw. Bleiglas) verwenden oder über ein großes Volumen verfügen. Daher ist die häufigste Bauform das **Sampling-Kalorimeter** (siehe Abbildung 1.7), das aus abwechselnden Lagen von Absorber- und Sensormaterial besteht. Innerhalb von Absorberschichten mit hoher Dichte werden dabei vermehrt Schauer gebildet, während die aktiven Detektorschichten ein Signal erzeugen, dass üblicherweise proportional zur Energie dieser Schauer und damit zur Energie des zu messenden Teilchens ist.

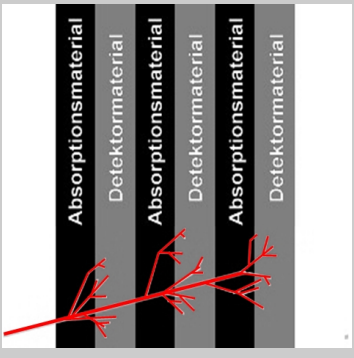


Abbildung 1.7: *Sampling-Kalorimeter[25].*

Kapitel 2

LHC und ATLAS

Nach der in Kapitel 1 gegebenen, kurzen Einführung in das Standardmodell der Teilchenphysik beschreibt das vorliegende Kapitel den im Rahmen dieser Doktorarbeit relevanten Beschleuniger LHC (2.1) und den ATLAS-Detektor (2.2).

2.1 Der LHC-Beschleuniger

Am europäischen Teilchenphysik-Zentrum CERN bei Genf befindet sich mit dem LHC der momentan größte und modernste Teilchenbeschleuniger der Welt. Zwischen dem Genfer See und den Bergen des Jura gelegen, nutzt der LHC-Beschleuniger den Tunnel des ehemaligen LEP-Beschleunigers, der sich durchschnittlich 100 m tief unter der Erdoberfläche befindet und einen Umfang von etwa 27 km hat. An vier Positionen des Rings existieren Wechselwirkungszonen, die jeweils einen Detektor beherbergen. Abbildung 2.1 zeigt eine schematische Darstellung des LHC-Beschleunigers, sowie der vier Detektoren ATLAS, ALICE[1], CMS[2] und LHCb[3].

Bei ATLAS und CMS handelt es sich um Vielzweckdetektoren, die das gesamte Spektrum der am LHC zugänglichen Physik abdecken. ALICE und LHCb hingegen sind Detektoren, deren Design auf die Untersuchung spezieller Teilbereiche der Hochenergiephysik ausgerichtet ist.

Der ALICE-Detektor untersucht die Eigenschaften des "Quark-Gluon-Plasmas", einer Materieform wie sie im Universum vermutlich Sekundenbruchteile nach dem Urknall das letzte Mal existiert hat. Dazu werden in speziellen LHC-Läufen Blei-Ionen mit einer Schwerpunktsenergie von bis zu 1150 TeV im Ring beschleunigt und im ALICE-Detektor

[1] **A** **L**arge **I**on **C**ollider **E**xperiment
[2] **C**ompact **M**uon **S**olenoid
[3] **L**arge **H**adron **C**ollider **b**eauty experiment

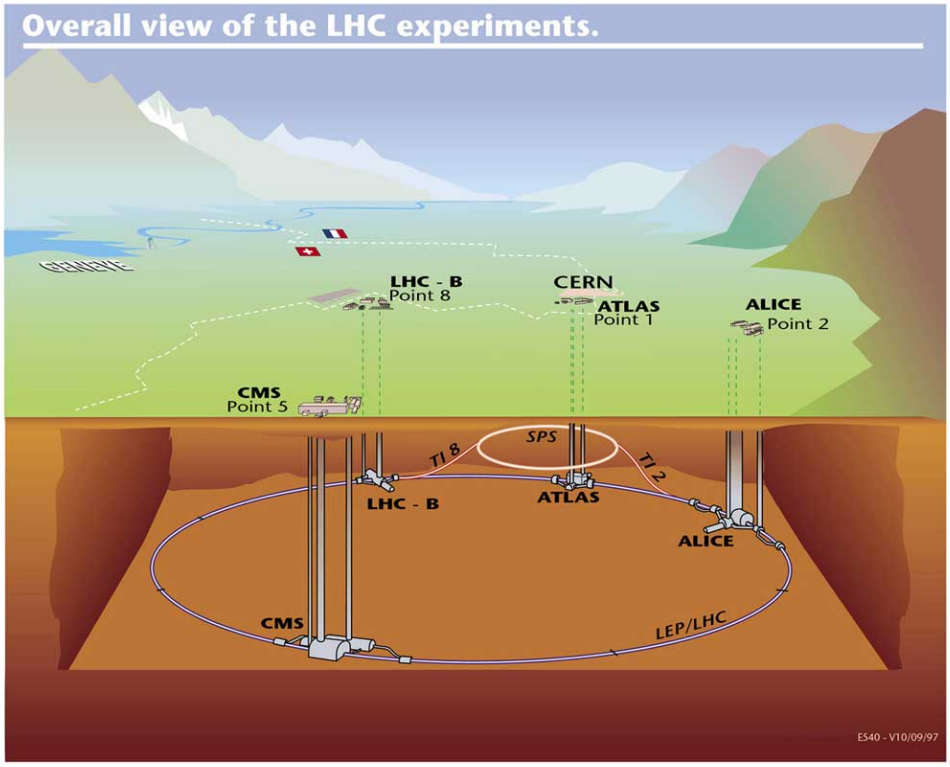


Abbildung 2.1: *Schematische Darstellung des LHC-Beschleunigers mit seinen vier großen Experimenten[3]*

zur Kollision gebracht. Auf Grund der hohen Energiedichte entsteht inmitten des Detektors für kurze Zeit ein Zustand ungebundener Quarks und Gluonen, den zu untersuchen das Ziel von ALICE ist.

LHCb wird versuchen, möglichst präzise Informationen über die Eigenschaften von b- und $\bar{b}$-Quarks zu sammeln. Insbesondere aus Untersuchungen der Zerfallscharakteristika der beiden Teilchen erhofft man sich wichtige Erkenntnisse über die beobachtete Asymmetrie zwischen Materie únd Antimaterie im Universum.

Mit einer Schwerpunktsenergie von bis zu 14 TeV wird der LHC die höchstenergetischen Teilchenkollisionen produzieren, die bislang im Labor hergestellt werden konnten. Da die Beschleunigung von Elektronen und Positronen auf diese hohen Energien bedingt durch Synchrotronstrahlungsverluste im ehemaligen LEP-Tunnel nicht möglich ist, werden im LHC Protonen beschleunigt. Gemäß der Beziehung 1.4 unterliegen diese einem um etwa den Faktor $16 \cdot 10^{12}$ geringeren Energieverlust. Die beim Umlauf der Protonen mit Kollisionsenergie (14 TeV) im LHC benötigte Leistung zur Kompensation der Synchrotronstrahlung beträgt weniger als 4 kW[5].

Weiterhin birgt die Beschleunigung von Protonen den Vorteil, dass die Quarks als Konstituenten des Protons zusätzlich zur elektromagnetischen und schwachen Wechselwirkung, an der ja auch geladene Leptonen partizipieren, an der starken Wechselwirkung teilnehmen. Auf Grund der großen Kopplungsstärke der starken Wechselwirkung ergeben sich in den meisten Fällen Wirkungsquerschnitte, die weit über denen der schwachen oder elektromagnetischen Kräfte liegen. Im Zusammenspiel mit der hohen Design-Luminosität von 10^{34} cm^{-2}s^{-1} ergeben sich hohe Produktionsraten für die verschiedensten, für die Hochenergiephysik interessanten Prozesse. Abbildung 2.2 zeigt die Wirkungsquerschnitte der am LHC dominanten Prozesse, sowie die bei Design-Luminosität zu erwartenden Raten.

Um die erwünschte Luminosität zu erreichen, ist es notwendig möglichst viele Teilchen pro Zeiteinheit miteinander in Wechselwirkung zu bringen. Der LHC wird mit insgesamt bis zu 2808 Teilchenbündeln, "Bunches" genannt, gefüllt, die jeweils etwa 10^{11} Protonen enthalten. Die einzelnen Pakete werden dabei von der Vorbeschleunigerkette des LHC erzeugt und in den LHC-Ring eingespeist. Diese besteht aus dem LINAC[4]-Linearbeschleuniger, sowie den Synchrotron-Vorbeschleunigern PS[5] und SPS[6]. Nach Durchlaufen des SPS besitzen die Protonen eine Energie von 450 GeV, mit der sie in den LHC injiziert werden.

An den Kollisionspunkten werden die Teilchenpakete von magnetischen Linsen auf ein zylindrisches Volumen von weniger als 20 μm Durchmesser und 8 cm Länge gebracht und treffen sich unter einem Winkel von ca. 300 μrad. Der Abstand zweier aufeinander folgender Bunches beträgt dabei in etwa 7,5 m, so dass sich bei den gegebenen, hochenergetischen Protonen alle 25 ns Bunches an den Wechselwirkungspunkten treffen[7].

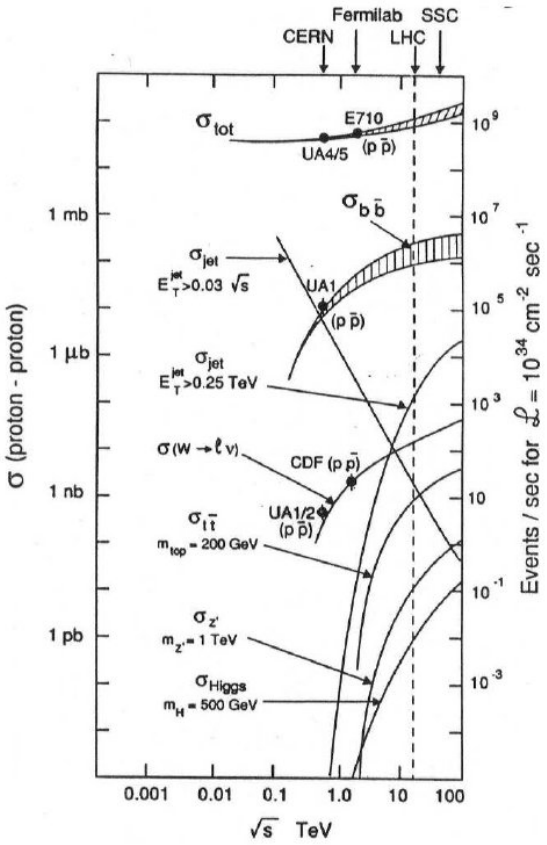


Abbildung 2.2: *Wirkungsquerschnitte und Raten der dominanten LHC-Prozesse[13]*

[4]**Lin**ear **Ac**cellerator
[5]**P**roton **S**ynchrotron
[6]**S**uper **P**roton **S**ynchrotron
[7]Eine gleichmäßige Verteilung der 2808 Bunches auf den 27 km umfassenden LHC-Ring hätte in etwas 9,5 m voneinander entfernte Bunches zur Folge. Die komplexe Vorbeschleunigerstruktur des LHC führt dazu, dass nur 2808 der 3564 möglichen Bunchplätze besetzt sind.

Die hohe Ring-Umlauffrequenz der Bunches von mehr als 11 kHz führt in Verbindung mit ihrer Anzahl zu etwa 32 Millionen Bunchcrossings pro Sekunde[8]. Da im Mittel etwa 22 Protonenpaare pro Bunchcrossing miteinander inelastisch wechselwirken, müssen die Detektoren in der Lage sein, ca. 600 Millionen inelastische Prozesse pro Sekunde zu verarbeiten[9].

Da die gewünschte Luminosität bei einem Proton-/Antiprotonbeschleuniger auf Grund der Schwierigkeit, Antiprotonen in ausreichend hoher Zahl bereitzustellen, nicht realisierbar war, entschied man sich, den LHC als Proton-/Protonbeschleuniger zu bauen. Im Gegensatz zu Teilchen-/Antiteilchenbeschleunigern, wie dem Tevatron am Fermilab[10] oder dem LHC-Vorgänger LEP, ist es damit allerdings nicht mehr möglich, die Teilchen in einem einzigen Strahlrohr gegenläufig zu beschleunigen. Die Konstruktion der 1232 Haupt-Dipolmagnete des LHC erlaubt es allerdings, das Magnetfeld für zwei etwa 20 cm voneinander getrennte Strahlröhren zu nutzen, in denen die Protonen gegenläufig umlaufen.

Ohne eine häufige Wiederherstellung der Strahlgeometrie würden die Protonen auf Grund von Strahl-Strahl-Wechselwirkung, Protonabstoßung innerhalb eines Bunches, Imperfektionen in den Dipol-Magnetfeldern und weiterer Effekte innerhalb kürzester Zeit die gewünschte Luminosität unterschreiten. Um das zu verhindern, besitzt der LHC eine komplexe, aus mehr als 6000 Quadru-, Sexto- und Oktopolmagneten[11] bestehende Magnetoptik, die in der Lage ist, die durch obige Effekte verursachten Probleme zu kompensieren. Während dieses aufwändige Magnetsystem in der Lage ist, unerwünschte Defokussierungseffekte zu minimieren, wird der Einsatz dieser großen Anzahl an Korrekturmagneten durch die effektive Verkleinerung des Beschleuniger-Radius erkauft. Denn der von den Korrekturmagneten eingenommene Platz in der Strahllaufbahn kann nicht von den, für die Ablenkung der Protonen auf eine Kreisbahn zuständigen Dipolmagneten genutzt werden. Für die Berechnung der notwendigen Magnetfeldstärke nach Gleichung 1.5 muss man daher den effektiven Krümmungsradius von ca. 2800 m einsetzen und nicht den tatsächlichen Radius des Beschleunigertunnels (immerhin etwa 4250 m).

Die für die Zirkulation von Protonen mit je 7 TeV Energie benötigte Magnetfeldstärke in den Dipolmagneten berechnet sich demnach zu rund $8,3$ T (pro 14 m langem Dipol). Magnetfelder dieser Größenordnung können nur durch den Einsatz supraleitender Ablenkmagnete erreicht werden. Um ihre supraleitenden Eigenschaften nicht zu verlieren, müssen die am LHC eingesetzten Magnete auf eine Temperatur von bis zu $1,9$ K heruntergekühlt

[8] Hier gilt dasselbe Argument: da nicht alle möglichen Bunchspaces besetzt sind, ist die naive Annahme von $\frac{1}{25\text{ ns}} = 40 \cdot 10^6\text{ s}^{-1}$ ungenau. Da die Anforderungen an die Instrumentierung in den meisten Fällen nicht von der mittlerer Kollisionszahl, bzw. dem Bunchspacing, sondern von den effektiven Maximalraten abhängen, werden diese oft angegeben.

[9] Näheres dazu im Abschnitt 2.2.

[10] Fermi National Accelerator Laboratory, Batavia, USA

[11] Tatsächlich sind am LHC sogar noch höhere Ordnungen von Magneten im Einsatz.

werden, was durch den Einsatz von suprafluidem Helium erreicht wird[12]. Das suprafluide Helium hat neben seiner geringen Temperatur weiterhin den Vorteil, eine sehr hohe Wärmeleitfähigkeit sowie eine verschwindende mechanische Reibung aufzuweisen. Die Verbindung dieser Vorteile führt dazu, dass sich suprafluides Helium sehr gut für den Einsatz als Kühlmittel für die LHC-Magnete eignet.

2.2 Das ATLAS-Experiment

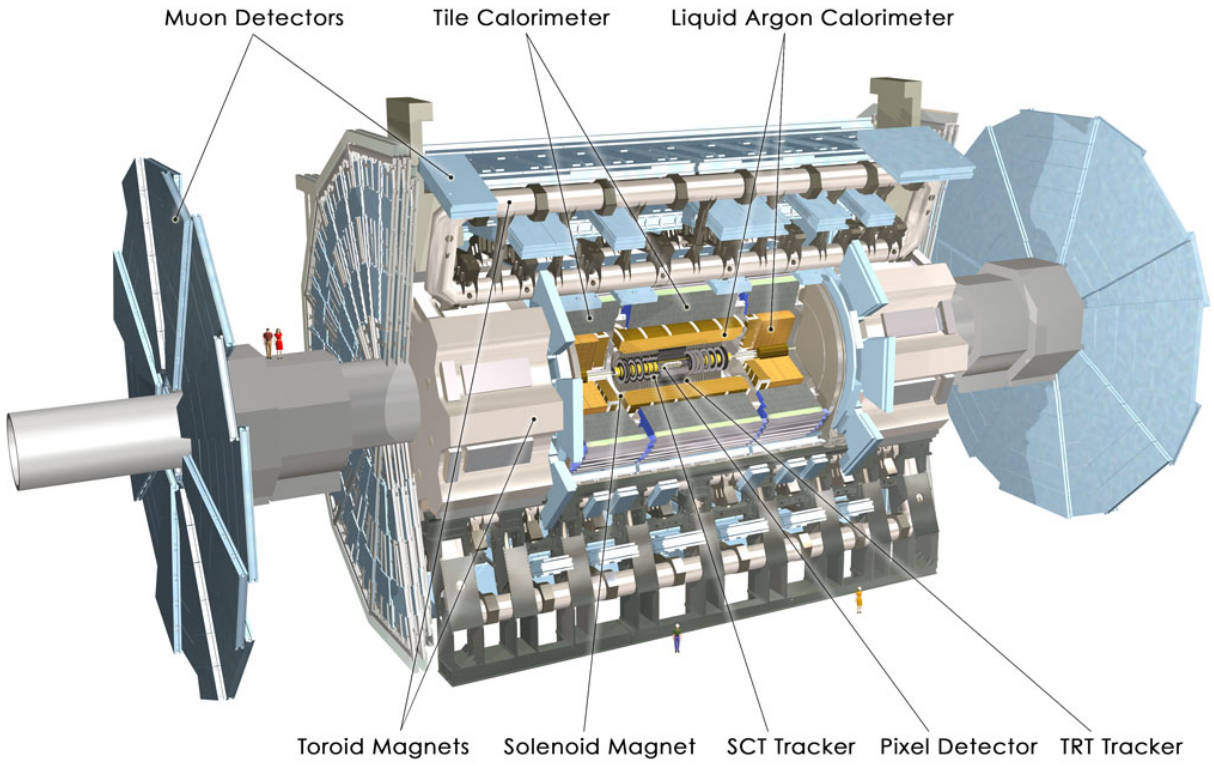


Abbildung 2.3: *Risszeichnung des ATLAS-Detektors und seiner Subdetektoren[3].*

Der ATLAS-Detektor ist neben dem bereits erwähnten CMS-Detektor einer von zwei Vielzweckdetektoren am LHC. Seine Aufgabe besteht darin, ein möglichst großes Spektrum an physikalisch interessanten Prozessen über den gesamten Parameterbereich des LHC beobachtbar zu machen.

[12]Suprafluides Helium gehört übrigens zu den wenigen makroskopischen, bosonischen Substanzen. Diese haben teilweise sehr eigenartige Eigenschaften, so sinkt die Viskosität des flüssigen Heliums unterhalb des "Lambda-Punktes" (für Helium etwa $2,2$ K) auf exakt Null. Das führt unter anderem dazu, dass die Flüssigkeit an Wänden hochkriechen kann (Rollins-Film).

Um dieses Ziel zu erreichen, besteht der ATLAS-Detektor (siehe Abbildung 2.3) aus einer Vielzahl unabhängiger Subdetektoren, die jeweils eine eigene Aufgabe übernehmen.

Mit einer Länge von etwa 44 m und einem Durchmesser von 22 m ist der zylindrische ATLAS-Detektor der größte Detektor des LHC. Trotz dieser enormen Größe, wiegt der Detektor lediglich 7000 t[13].

Möglich ist dieses vergleichsweise geringe Gewicht durch den Einsatz einer für Detektoren ungewöhnlichen Magnetfeldanordnung. Der in Abschnitt 2.2.1 näher beschriebene innere Detektor wird dabei durch ein konventionelles, solenoides (also zur Strahlachse paralleles) 2T-Magnetfeld in die Lage versetzt, Teilchenimpulse genau zu rekonstruieren. Um auch die Impulse der hochenergetischen Myonen messen zu können, die den kompletten Detektor zu durchqueren vermögen, kommt außerhalb des inneren Detektors ein toroidales Magnetfeld von $0,5$ T bis 1 T Stärke zum Einsatz, dessen Feldlinien die Strahlachse ringförmig umgeben (siehe Abbildung 2.4). Diese Anordnung hat neben der, durch die vergleichsweise leicht-bauenden und wenig Platz einnehmenden Toroidspulen bedingten Platz und Materialeinsparung weiterhin den Vorteil, dass die auf die Teilchen wirkende Lorentzkraft unabhängig von der Pseudorapidität (Begriffserklärung siehe Kasten "ATLAS-Koordinatensystem") der Teilchen wird[14].

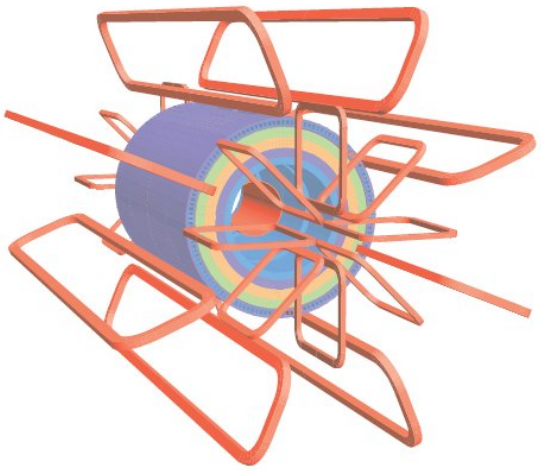


Abbildung 2.4: *Vereinfachte Darstellung des ATLAS-Magnetsystems [3]. In der Mitte des Detektors ist die tonnenförmige Solenoidspule des inneren Detektors (rot) zu erkennen, die im Bereich des Myonendetektors (hier nicht eingezeichnet) von den $3 \cdot 8$ Spulen des Toroidmagnetsystems umgeben ist.*

Funktional kann man den ATLAS-Detektor dabei in den bereits erwähnten inneren Detektor, das aus elektromagnetischem und hadronischem Kalorimeter bestehende Kalorimetersystem und den Myondetektor einteilen. Da alle Komponenten für die Rekonstruktion der am Wechselwirkungspunkt stattfindenden Prozesse wichtig sind, sollen sie in den folgenden Abschnitten erklärt werden. Die ATLAS-Subdetektoren umgeben den primären Wechselwirkungspunkt zwiebelschalenförmig, weshalb es sich anbietet, bei der

[13] Damit besitzt der ATLAS-Detektor eine geringe Dichte: würde man ihn luftdicht in eine Folie einschweißen, so könnte er im Wasser schwimmen.

[14] In einem solenoiden Magnetfeld wird ein Teilchen hoher Rapidität auf Grund der bezüglich des Magnetfelds kleinen senkrechten Geschwindigkeitskomponente nur eine geringe Lorentzkraft erfahren, was für die Impulsauflösung nachteilig ist.

Beschreibung der einzelnen Teildetektoren der Flugbahn der Teilchen, ausgehend von ihrem Vertex, zu folgen.

ATLAS-Koordinatensystem

Das offizielle ATLAS-Koordinatensystem ist so definiert, dass die Z-Achse entlang des Strahls verläuft, sich der Ursprung am Wechselwirkungspunkt befindet und die positive X-Achse auf den Ringmittelpunkt des LHC zeigt. Die durch ein rechtshändiges Koordinatensystem definierte, positive Z-Achse zeigt auf die "A-Seite" des Detektors, die negative auf die "C-Seite".

In Anbetracht der zylindrischen Geometrie sind Polarkoordinaten nützlich. Dabei bezeichnet ϕ den Azimutalwinkel um die Strahlachse und θ den Polarwinkel. R ist je nach Zusammenhang der Abstand vom Ursprung oder von der Strahlachse.

Bei der Beschreibung der Detektorgeometrie erweist sich die als "Pseudorapidität" bekannte Größe $\eta = -\ln\tan(\theta/2)$ als besonders wichtig, da die Detektoren zumeist sowohl rotationssymmetrisch in ϕ als auch spiegelsymmetrisch in Z sind.

Die bei der Beschreibung der Spurdetektoren und der Kalorimeter verwendeten Größen p_T und E_T beziehen sich auf die Projektionen von Impuls und Energie auf die X-Y-Ebene. Im Bereich der Kalorimeter wird weiterhin häufig die Größe $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$ als Maß für die Granularität der Detektoren verwendet.

2.2.1 Der innere Spurdetektor

Der innere Detektor (siehe Abbildung 2.5) des ATLAS-Experiments besteht aus den drei unabhängigen Spurdetektoren Pixeldetektor, SCT[15] und TRT[16]. Ihre gemeinsame Aufgabe ist die Vermessung der Teilchenimpulse mit einer Präzision von $\sigma_{p_T}/p_T = 0.05\% p_T \oplus 1\%$ bei gleichzeitig möglichst geringem Energieverlust der durchdringenden Teilchen. Darüberhinaus liefern der Pixeldetektor und TRT noch weitere wertvolle Informationen für die Rekonstruktion des aufgenommenen Ereignissen, auf die in den entsprechenden Abschnitten eingegangen wird.

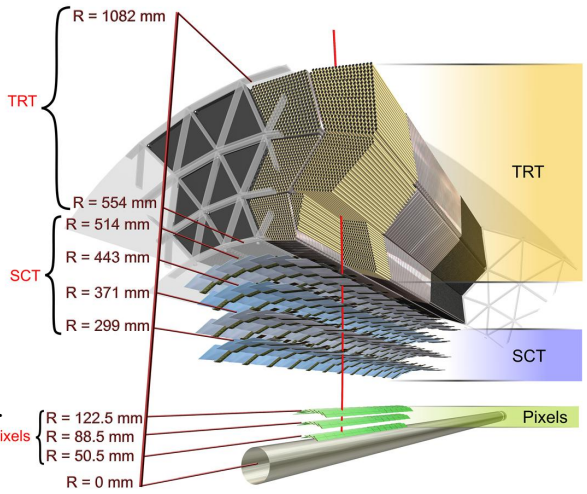


Abbildung 2.6: *Ausschnitt des Zentralbereichs des inneren Detektors mit angedeuteter Teilchenspur[3].*

[15] **S**emi **C**onductor **T**racker
[16] **T**ransition **R**adiation **T**racker

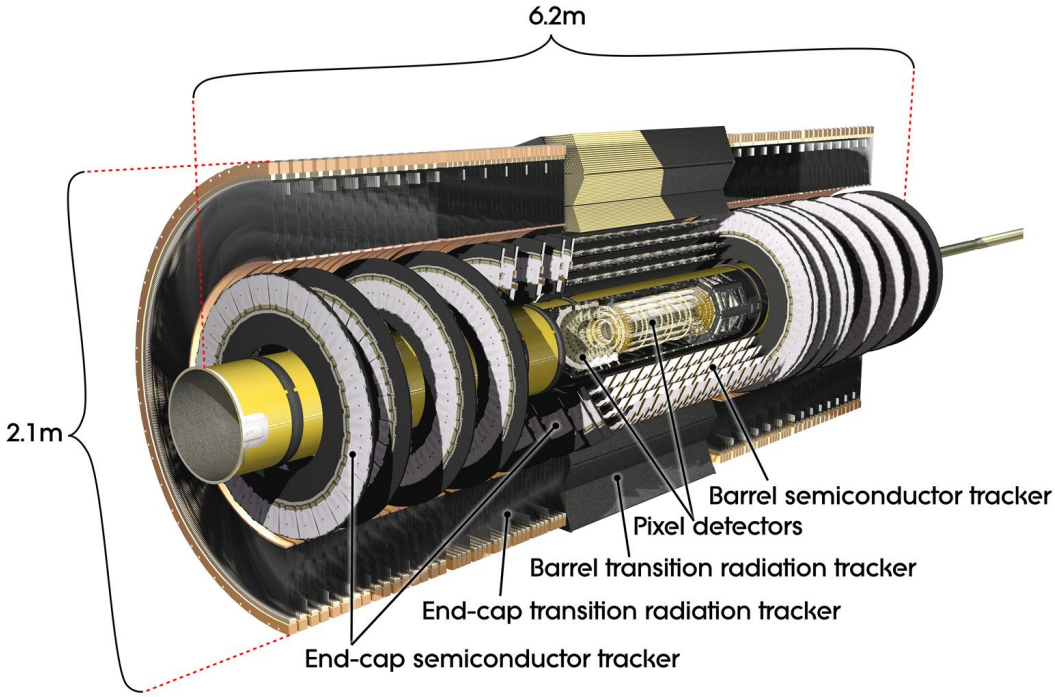


Abbildung 2.5: *Risszeichnung des inneren Tracking-Detektors und seiner drei Teildetektoren[3].*

Das Messprinzip der Spurdetektoren ist dabei immer das gleiche: geeignete Sensoren erfassen den Durchgang eines Teilchens und ermitteln dessen Koordinaten. Die zunächst als unabhängige Messpunkte aufgezeichneten "Hits" werden dann von speziellen Computerprogrammen, den "Tracking-Algorithmen" ausgewertet. Die Hauptaufgabe dieser Algorithmen ist es, aus einer gegebenen Ansammlung von Hits zusammenhängende (und durch die Hits gestützte) Flugbahnen der Teilchen zu rekonstruieren (siehe Abbildung 2.6). Aus der Parametrisierung der gewonnenen Flugbahnen lassen sich dann wichtige Informationen über das Teilchen ableiten. Geladene Teilchen werden vom solenoiden Magnetfeld des inneren Detektors auf Kreisbahnen[17] abgelenkt, deren Krümmungsradius bei Kenntnis der Magnetfeldstärke Auskunft über den Impuls des Teilchens gibt.

Die Anzahl der pro Zeiteinheit zu registrierenden inelastischen Primärwechselwirkungen ist auf Grund des hohen totalen inelastischen Wirkungsquerschnitts sowie der hohen Luminosität des LHC (siehe Abschnitt 2.1) sehr groß. Zusammen mit dem unvermeidlichen QCD-Untergrund eines Hadronenbeschleunigers ergeben sich hohe Teilchenflussdichten im inneren Detektor (siehe Abbildung 2.8). Da alle Sensoren eine gewisse Zeit nach Registrierung eines Teilchendurchgangs nicht für die Registrierung weiterer Teilchen genutzt werden können (Totzeit), muss man gewährleisten, dass ein solcher "Doppeltreffer" nur sehr selten

[17] Diese sind allerdings bei genügend hohem Impuls nicht mehr in sich geschlossen; die Teilchen werden lediglich auf einen Kreisbogen gezwungen, bevor sie das Magnetfeld verlassen.

vorkommt, indem man die Größe des Sensors entsprechend klein wählt. Weiterhin ist der absolute Teilchenfluss innerhalb des Solenoiden bezüglich des Radius relativ konstant[18], weshalb die Teilchenflussdichte mit abnehmendem Radius ansteigt (siehe Abbildung 2.7). Um dies zu kompensieren, müssen die physikalischen Abmessungen der von den Subdetektoren genutzten Sensorzellen mit abnehmendem Abstand vom Wechselwirkungspunkt immer kleiner werden.

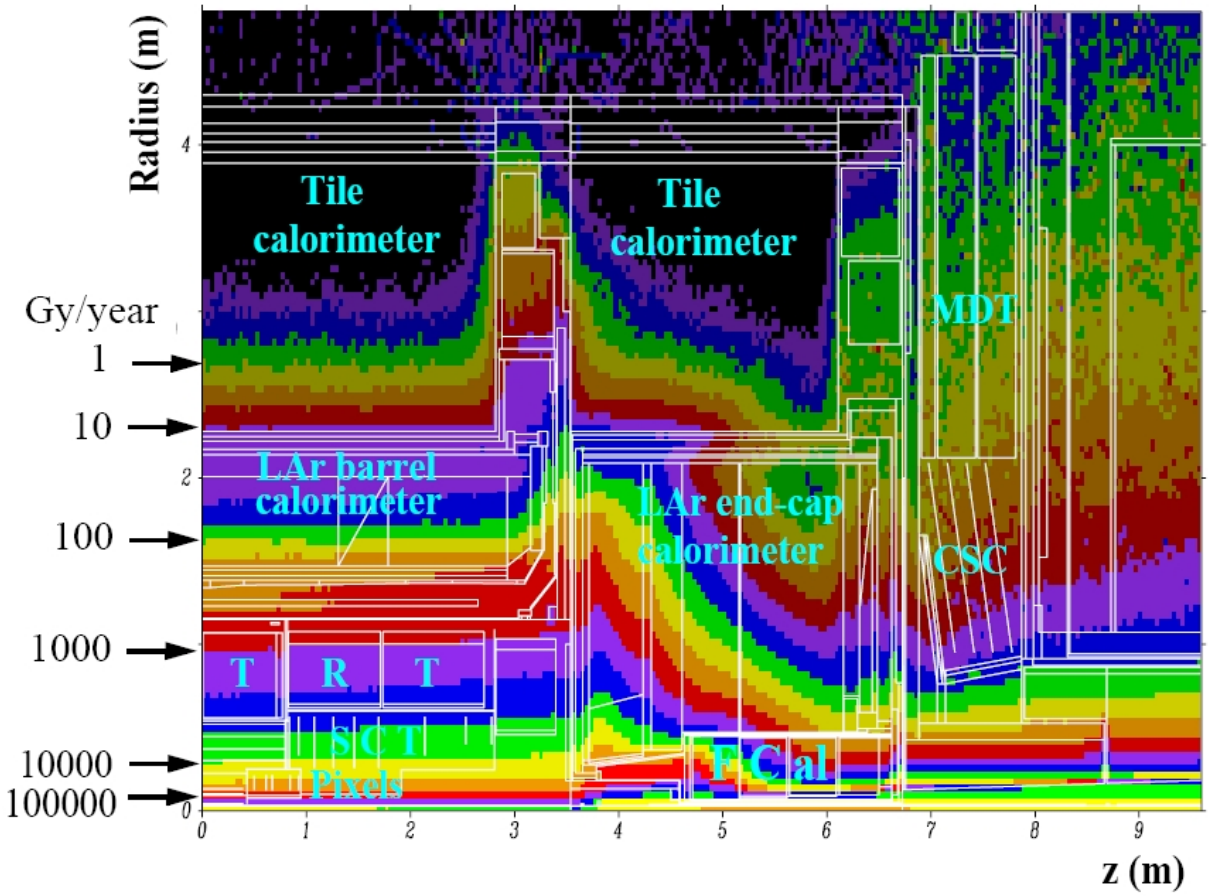


Abbildung 2.7: *Totale ionisierende Dosis pro Jahr[3].*

Die drei Subdetektoren des inneren Detektors umgeben den Wechselwirkungspunkt dabei zylinderschalenförmig. Die Dimensionen der einzelnen Detektoren sind in Tabelle 2.1 angegeben.

[18]Selbstverständlich ist der Teilchenfluss auch hier nicht konstant, da es bereits im inneren Detektor durch Paarbildung und Hadronisierung zu einem Anstieg der Teilchenzahl kommt. Im Vergleich zu den in den Kalorimetern provozierten Effekten ist diese jedoch gering.

Region	R (cm)	Particle rates (kHz/cm^2)						F_{neq} ($\times 10^{+12}$ cm^{-2})	Ionisation dose (Gy/y)
		γ	Protons	Neutrons	$\pi^{\pm}$	$\mu^{\pm}$	e^-		
		> 30 keV	> 10 MeV	> 100 keV	> 10 MeV	> 10 MeV	> 0.5 MeV		
Pixel layer 0	5.05	45800	2030	4140	34100	300	8140	270	158000
Pixel layer 2	12.25	9150	280	1240	4120	190	1730	46	25400
SCT barrel layer 1	29.9	4400	80	690	990	130	690	16	7590
SCT barrel layer 4	51.4	3910	36	490	370	67	320	9	2960
SCT end-cap disk 9	43.9	7580	73	840	550	110	470	14	4510
TRT outer radius	108.0	2430	10	380	61	7	53	5	680

Abbildung 2.8: *Teilchenflussraten und Dosisleistungen für die Subsysteme des inneren Detektors bei LHC-Design-Luminosität (10^{34} $cm^{-2}s^{-1}$)[3].*

2.2.1.1 Der Pixeldetektor

Der Pixeldetektor ist der innerste Subdetektor des ATLAS-Experiments. Als Teil des inneren Detektors besteht seine Aufgabe in der Aufzeichnung von mindestens drei Raumpunkten pro geladenem Teilchen. Der Detektor registriert geladene Teilchen dabei durch Messung der von ihnen freigesetzten Elektronen in segmentierten Halbleiterdioden. Obwohl auf die Details des Aufbaus des Pixeldetektors in Kapitel 3 eingegangen werden soll, sei hier erwähnt, dass seine innerste Lage lediglich etwa 5 cm von der Strahlachse entfernt ist. Um die gleichzeitige Besetzung ("Occupancy") der Sensorzellen möglichst gering zu halten und eine hohe Ortsauflösung zu erreichen, wurden für den Pixeldetektor pixelförmige Siliziumsensoren mit einer Länge (entlang der Strahlachse) von 400 μm (entlang Z bzw. R) und einer Breite von 50 μm (in "ϕ") verwendet, die dem Detektor seinen Namen geben. Die daraus resultierende hohe Ortsauflösung (siehe Tabelle 2.1) ermöglicht die genaue Zurückverfolgung der Teilchenflugbahnen zu ihrem jeweiligen Ursprung. Dieser Ursprung kann mit einer Genauigkeit von bis zu etwa 10 μm rekonstruiert werden, was es ermöglicht, kurzlebige Teilchen auf Grund ihres vom primären Vertex wenige mm entfernten sekundären Vertizes zu identifizieren. Mit über 80 Millionen Auslesekanälen stellt der Pixeldetektor trotz seiner geringen Abmessungen von 130 cm Länge und 30 cm Durchmesser mehr als 90% der ATLAS-Auslesekanäle.

2.2.1.2 Der Semi-Conductor-Tracker

Wie der Name bereits andeutet, handelt es sich auch beim zweiten von den Teilchen passierten Subdetektor um einen Siliziumdetektor. Während der SCT-Detektor genau wie der Pixeldetektor Halbleitersensoren zum Nachweis geladener Teilchen nutzt, besteht dieser aus sehr viel längeren Sensorzellen mit Dimensionen von 10 cm bis 12 cm Länge (wieder in Z oder R) mal 60 μm bis 90 μm Breite (in "ϕ"), die daher auch als "Streifen" (Strips) be-

	Radius [mm]	**Länge [mm]**	**Genauigkeit [μm]**	
Innerer Detektor	$0 \leq R \leq 1150$	$0 \leq \|z\| \leq 3512$	$R-\phi$	z/R
Strahlrohr	$29 \leq R \leq 36$			
Pixeldetektor	$45.5 \leq R \leq 242$	$0 \leq \|z\| \leq 3092$		
Zylinderlagen	$50.5 \leq R \leq 122.5$	$0 \leq \|z\| \leq 400.5$	10	115
Endkappen	$88.8 \leq R \leq 149.6$	$495 \leq \|z\| \leq 650$	10	115
SCT	$251 \leq R \leq 610$	$0 \leq \|z\| \leq 2797$		
Zylinderlagen	$299 \leq R \leq 514$	$0 \leq \|z\| \leq 749$	17	580
Endkappen	$275 \leq R \leq 560$	$839 \leq \|z\| \leq 2735$	17	580
TRT	$554 \leq R \leq 1106$	$0 \leq \|z\| \leq 2744$		
Zylinderlagen	$563 \leq R \leq 1082$	$0 \leq \|z\| \leq 780$	130	
Endkappen	$644 \leq R \leq 1004$	$848 \leq \|z\| \leq 2710$	130	

Tabelle 2.1: *Hauptparameter der drei Subdetektoren des inneren Detektors[3].*

zeichnet werden. Dies ist einerseits möglich, da die Teilchenflussdichte im Bereich des SCT bereits stark genug abgefallen ist, andererseits notwendig, da der SCT, mit einer Länge von fast 2 m und einem Durchmesser von mehr als 1 m eine wesentlich größere Oberfläche besitzt als der Pixeldetektor und der Einsatz von Pixelsensoren in diesem Maßstab extrem teuer geworden wäre. Auf Grund der geringen Streifenbreite besitzt der SCT in ϕ eine ähnlich gute Ortsauflösung wie der Pixeldetektor. Um trotz der eingesetzten Streifensensoren eine möglichst gute Ortsauflösung in Z zu erreichen, sind je zwei hintereinander liegende Streifen leicht gegeneinander verkreuzt (der Kreuzungswinkel beträgt dabei etwa 160 μrad bis 210 μrad), so dass sich aus den Trefferinformationen der beiden getroffenen Streifen die Längskoordinate berechnen lässt. Die daraus resultierende Auflösung liegt bei immerhin etwa 580 μm (siehe Tabelle 2.1).

Der SCT liefert mit vier zylindrischen Lagen im Zentralbereich und je neun scheibenförmigen "Disks" im Bereich der Endkappen mindestens vier weitere Raumpunkte für die erwähnten Spurrekonstruktions-Algorithmen.

2.2.1.3 Der Transition-Radiation-Tracker

Der TRT ist der äußerste Subdetektor des inneren Detektors. Mit einer Länge von über 5 m und einem Durchmesser von mehr als 2 m ist eine Instrumentierung des benötigten Volumens mit Hilfe von Halbleitersensoren sehr aufwendig. Daher nutzt der TRT "Straw Tubes", bei denen es sich um 4 mm durchmessende, gasgefüllte[19] Driftkammern handelt. Im Zentralbereich des Detektors werden dabei zweimal 144 cm lange, parallel zur Strahlachse verlaufende Tubes eingesetzt, wohingegen in den Endkappen 37 cm lange, radiale

[19] $70\%Xe, 27\%CO_2, 3\%O_2$

Röhrchen zum Einsatz kommen. Die insgesamt etwa 170000 Straw Tubes sind in mindestens 36 (Barrel) und maximal 80 (Disk) Lagen angeordnet, wobei die Driftröhren im Zentralbereich in je zwei[20] Auslesekanäle unterteilt sind, um die Besetzung zu reduzieren. Der TRT-Detektor erreicht pro Straw eine Auflösung von 130 μm in $R-\Phi$ und kann damit, auf Grund der Vielzahl der über ein relativ großes Volumen aufgenommenen Raumpunkte, wertvolle Beiträge zur Implusrekonstruktion liefern. Da der TRT, abgesehen von der oben erwähnten Segmentierung, keine Aussagen über die Teilchenposition entlang der Straws machen kann, ist es in jedem Fall notwendig, seine Informationen mit denen der anderen Spurdetektoren zu kombinieren.

Eine weitere Besonderheit des TRT ist die Fähigkeit, "Übergangsstrahlung" beim Durchgang hochrelativistischer Teilchen zu detektieren. Diese tritt immer dann auf, wenn ein Teilchen eine dielektrische Grenzfläche passiert, weshalb zwischen den Straw Tubes entsprechende Radiator-Materialien eingebaut wurden. Im Barrelbereich sind dies Polypropylen-Fasern, während im Endkappenbereich Polypropylen-Folien als Raditor verwendet werden. Die dabei abgestrahlten Photonen werden vom Gasgemisch des TRT als in der Stärke deutlich von Tracking-Treffern unterscheidbares Signal sichtbar gemacht. Zwei unterschiedliche "Low"- und "High"-Schwellen erlauben die Unterscheidung zwischen Spurelement und Übergangsstrahlung pro Auslesekanal. Da die Intensität I der Übergangsstrahlung mit steigender Teilchenmasse m gemäß

$$I \propto \gamma = \frac{E}{m} \tag{2.1}$$

abfällt [32](E ist hier die Teilchenenergie), ist der TRT als einziger Spurdetektor in der Lage, Elektronen auf Grund ihrer stärkeren Übergangsstrahlung von anderen geladenen Teilchen zu unterscheiden.

2.2.2 Die Kalorimeter

Das Kalorimetersystem des ATLAS-Detektors (siehe Abbildung 2.9) besteht aus einem elektromagnetischen und einem hadronischen Kalorimeter. Die einzelnen Komponenten der beiden Systeme werden in den folgenden Abschnitten beschrieben, während Tabelle 2.2 den detailierten Aufbau des Kalorimetersystems wiedergibt.

2.2.2.1 Das Elektromagnetische Kalorimeter

Das elektromagnetische Kalorimeter des ATLAS-Detektors besteht aus mehreren Teildetektoren, die zusammen genommen einen Bereich $|\eta| \leq 4.9$ abdecken. Die geforder-

[20] Tatsächlich sind die Straw Tubes in den innersten Barrel-Lagen in je drei Kammern unterteilt, wobei die mittlere Kammer wegen der hohen Besetzung nicht genutzt wird.

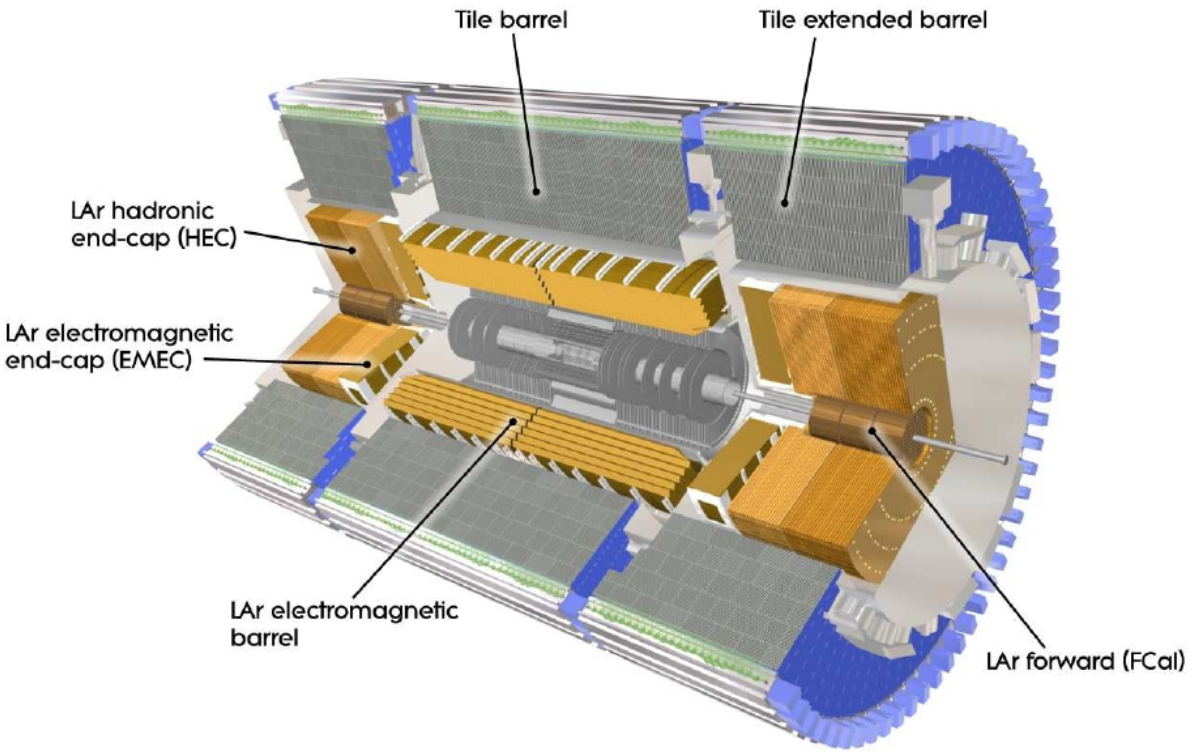


Abbildung 2.9: *Risszeichnung der ATLAS-Kalorimeter[3]*

te und inzwischen durch Messungen am Teststrahl bestätigte Energieauflösung des EM-Kalorimeters beträgt $\sigma_E/E = 10\%/\sqrt{E} \oplus 0.7\%$.

Alle EM-Kalorimeter sind dabei in Samplingbauweise[21] aufgebaut und nutzen wegen seiner guten Verstärkungseigenschaften und seiner intrinsischen Strahlenhärte flüssiges Argon als sensitives Detektormaterial. Je nach Anforderungen an die Strahlentoleranz und Granularität wird dabei jedoch das Absorbermaterial, sowie dessen Geometrie den Umständen angepasst.

So kommt im Zentralbereich eine akkordeonartig aufgebaute Bleiabsorberstruktur (hermetische Überdeckung in ϕ) zum Einsatz die sich innerhalb des Kryostaten, der für die Kühlung des Solenoiden des inneren Detektors benötigt wird, befindet. Durch diese Anordnung wird im Präzisionmessbereich des LAr[22]-EM-Barrel-Kalorimeters ($|\eta| \leq 1.475$) gewährleistet, dass lediglich ein Minimum an Strahlungslänge (und damit Energieverlust) vor dem Kalorimeter eingebaut ist (siehe auch Abbildung 3.1). Das elektromagnetische

[21] Als Samplingbauweise bezeichnet man die abwechselnde Anordnung aus Absorber- und Sensormaterial, siehe auch Kasten in Abschnitt 1.3.2.

[22] **L**iquid **Ar**gon

Barrelkalorimeter umgibt den Zentralbereich des inneren Detektors (genaue Maße finden sich in Tabelle 2.2).

Im Vorwärtsbereich $1.375 \leq |\eta| \leq 3.2$ kommen die elektromagnetischen Endkappen-Kalorimeter EMEC[23] zum Einsatz, welche sich sich zusammen mit den hadronischen Endkappenkalorimetern und den FCal[24]-Vorwärtskalorimetern in den Liquid-Argon-Kryostaten der Endkappen befinden.

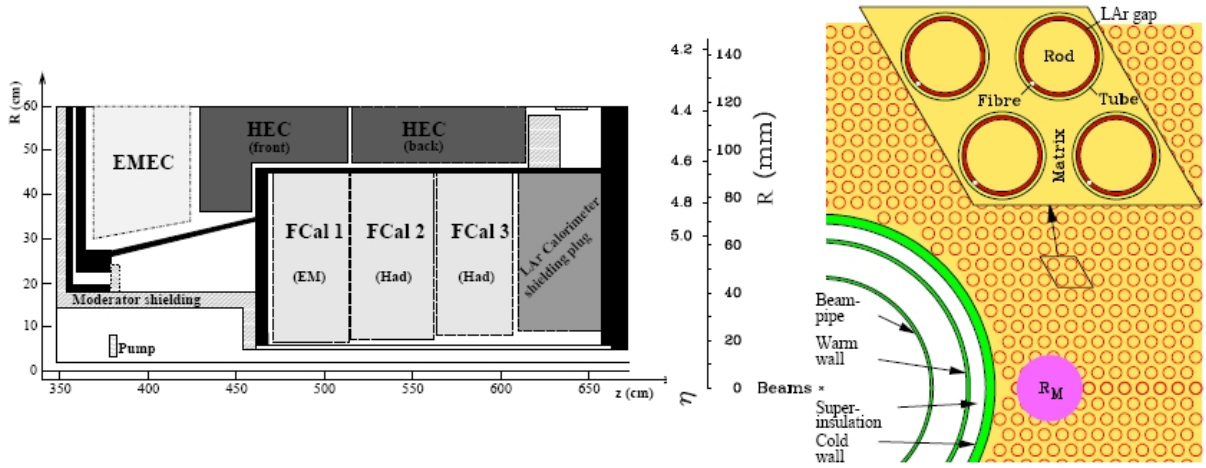


Abbildung 2.10: *Risszeichnung des Vorwärtskalorimeters (FCal)[3]. Links ist die Anordnung der drei FCal Subdetektoren FCal1, FCal2 und FCal3 entlang eines Längsschnittes des Detektors abgebildet. Die rechte Abbildung zeigt einen Querschnitt des FCal1-Kalorimeters. Im vergrößerten Ausschnitt ist die Kupfermatrix mit den in den Bohrlöchern zentrierten Kupferanoden ("Rods") und dem umgebenden LAr-Mantel ("Gap") sichtbar.*

Jenseits von $3.1 \leq |\eta|$ übernimmt dann das elektromagnetische FCal-Vorwärtskalorimeter bis zu einer Pseudorapidität von $|\eta| \leq 4.9$. Im Gegensatz zum Zentral- und Endkappenbereich mit der erwähnten Akkordeongeometrie kommt im FCal1, auf Grund der benötigten Strahlenhärte, eine Matrixanordnung von mit flüssigem Argon gefüllten Bohrlöchern in Kupferabsorbern zum Einsatz (siehe Abbildung 2.10).

Die Granularität des EM-Kalorimeters variiert dabei zwischen 0.025×0.025 (Barrel) und 0.1×0.1 (EMEC) in $\Delta\eta \times \Delta\phi$. Zudem werden je 4×4 Kalorimeterzellen zu "Trigger-Towern" zusammengefasst, deren Bedeutung in Abschnitt 2.2.4 erläutert wird.

2.2.2.2 Das Hadronische Kalorimeter

Wie man ebenfalls auf Abbildung 2.10 sehen kann, besteht das Vorwärtskalorimeter nicht nur aus dem elektromagnetischen FCal1, sondern auch aus den beiden hadronischen FCal2-

[23] **E**lectro **M**agnetic **E**nd **C**ap
[24] **F**orward **Cal**orimeter

	Radius [m]	**Länge [m]**	**Pseudorapidität**
Elektromagn. Kal.			
EM-Barrel	$1.4 \leq R \leq 2$	$0 \leq \|z\| \leq 3.2$	$\|\eta\| \leq 1.475$
EMEC	$0.33 \leq R \leq 2$	$3.7 \leq \|z\| \leq 4.3$	$1.375 \leq \|\eta\| \leq 3.2$
FCal1	$0.07 \leq R \leq 0.46$	$4.7 \leq \|z\| \leq 5.15$	$3.1 \leq \|\eta\| \leq 4.9$
Hadronisches Kal.			
Tile	$2.28 \leq R \leq 4.25$	$0 \leq \|z\| \leq 6.1$	$\|\eta\| \leq 1.7$
HEC	$0.5 \leq R \leq 2$	$4.3 \leq \|z\| \leq 6.1$	$1.5 \leq \|\eta\| \leq 3.2$
FCal2, FCal3	$0.07 \leq R \leq 0.46$	$5.2 \leq \|z\| \leq 6.05$	$3.1 \leq \|\eta\| \leq 4.9$

Tabelle 2.2: *Aufbau des ATLAS-Kalorimetersystems[3].*

und FCal3-Kalorimetern. Diese basieren auf einer dem FCal1 ähnlichen Matrixbauweise aus Wolframabsorbern und, wie alle hadronischen Kalorimeter des ATLAS-Detektors, auf dem Funktionsprinzip des Sampling-Kalorimeters. Um die "Albedo"[25] zu reduzieren, wurde das gesamte Vorwärtskalorimeter um etwa 1.2 m im Vergleich zu EMEC zurückversetzt, was zusätzlich eine hermetische Überdeckung der einzelnen Kalorimeter ermöglicht (siehe Abbildung 2.10).

Zwischen dem EMEC und den FCal-Vorwärtskalorimetern befinden sich die hadronischen Endkappenkalorimeter HEC[26], die auf jeder Seite in zwei hintereinander (in Z) aufgereihte Räder, HEC1 (Front) und HEC2 (Back) aufgeteilt sind. Beide Räder haben einen Außendurchmesser von 4 m und bestehen aus Kupferabsorbern mit flüssigem Argon als aktivem Nachweismaterial.

Im Zentralbereich des hadronischen Kalorimeters ($|\eta| \leq 1.7$) kommen "Tile"-Kalorimeter zum Einsatz. Anders als bei allen anderen bei ATLAS eingesetzten Kalorimetern setzen diese nicht auf flüssiges Argon als aktives Detektormaterial, sondern nutzen Plastikszintillatoren, die zwischen Stahlabsorberplatten eingebaut sind. Im Gegensatz zu der direkten elektrischen Auslese der LAr-Kalorimeter kommen hier über Lichtleiter mit den Szintillatoren verbundene Photomultiplier zum Einsatz (siehe Abbildung 2.11). Das Tile-Kalorimeter besteht aus drei getrennten Kalorimeterein-

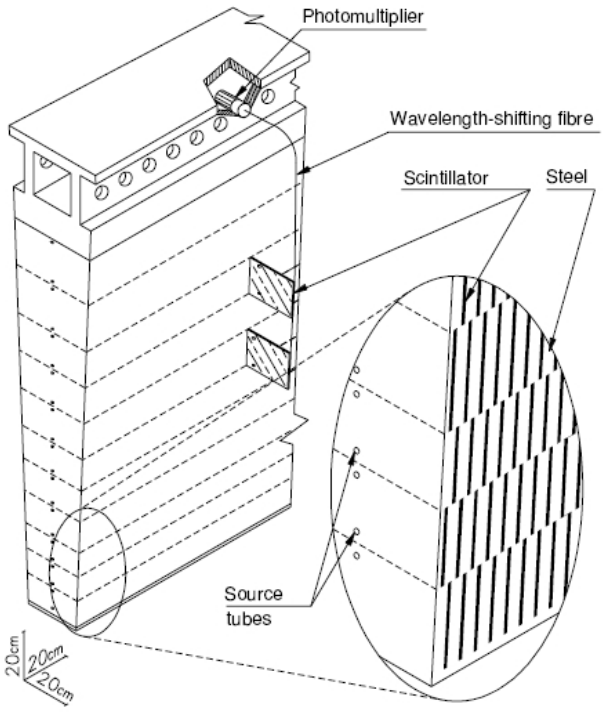


Abbildung 2.11: *Schematische Darstellung der optischen Auslese des Tile-Kalorimeters[3].*

[25] das "Rückstrahlen" von Kalorimeter-Neutronen in den Bereich des Inneren Detektors

[26] **H**adronic **E**nd **C**ap

heiten, dem zentralen Barreldetektor mit einer Länge von 5.8 m, sowie je ein erweiterter Barreldetektor auf A- und C-Seite mit einer Länge von 2.6 m. Die mechanische Struktur des Tile-Kalorimeters besteht aus tortenstückförmigen "Wedges", von denen eines schematisch in Abbildung 2.11 gezeigt ist. Das untere Ende der Grafik zeigt dabei zur Strahlachse. Je 64 dieser Wedges formen je eines der drei Tile-Kalorimeter. Die mechanische Aufhängung am äußeren Rand der Wedges beherbergt nicht nur die Ausleseelektronik und die Photomultiplier, sondern dient auch als Rückflussjoch für das Magnetfeld des Solenoiden.

Die Energieauflösung der hadronischen Kalorimeter liegt bei $\sigma_E/E = 50\%/\sqrt{E} \oplus 3\%$ für die Tile- und HEC-Kalorimeter, sowie $\sigma_E/E = 100\%/\sqrt{E} \oplus 10\%$ für die FCal3- und FCal3-Vorwärtskalorimeter und einer Granularität von 0.1×0.1 in $\Delta\eta \times \Delta\phi$.

2.2.3 Der Myondetektor

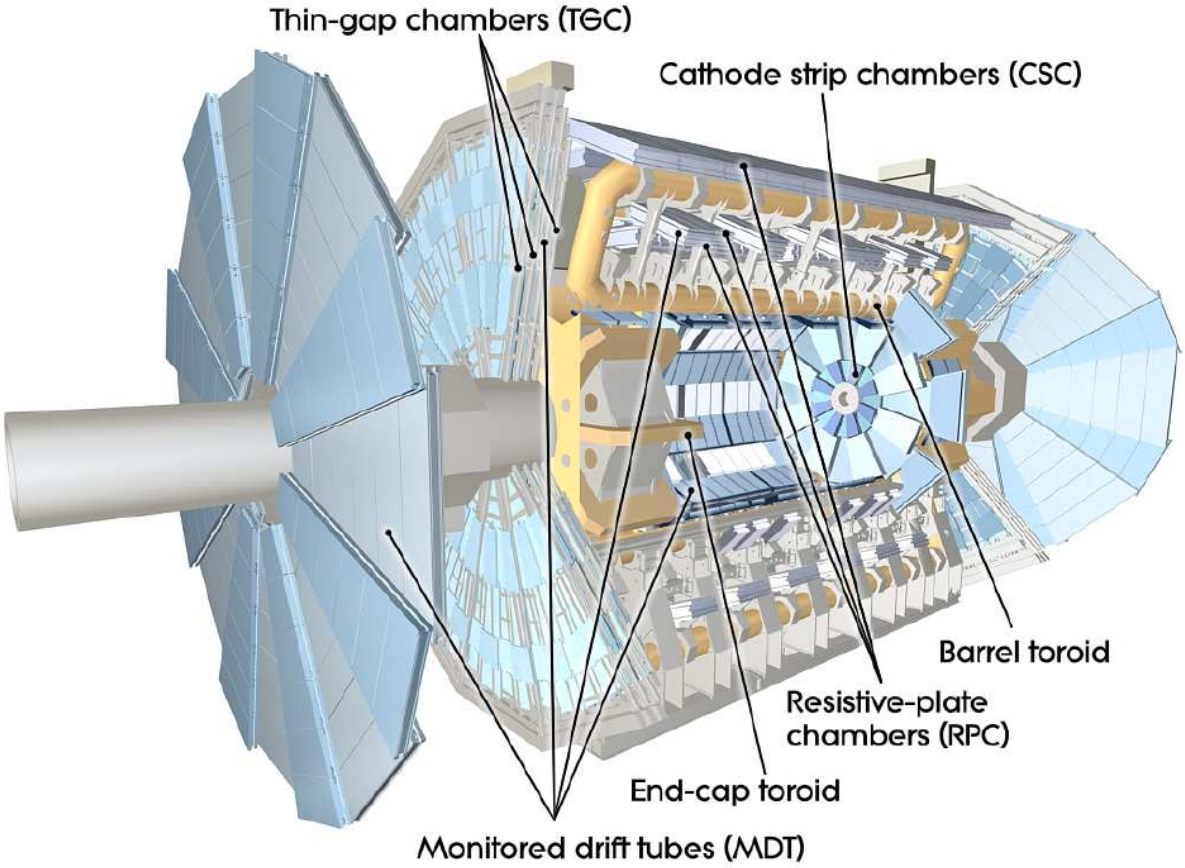


Abbildung 2.12: *Risszeichnung der ATLAS Myondetektors[3]*

Der Myondetektor ist, bezogen auf das instrumentierte Volumen, der größte Subdetektor des ATLAS-Experiments. Mit einem maximalen Durchmesser von etwa 22 m und einem

maximalen Abstand von 21.5 m vom Wechselwirkungspunkt in Z, definiert der Myondetektor die Grunddimensionen von ATLAS. Ähnlich dem TRT nutzen seine Subsysteme die Gasverstärkung in Proportionalkammern oder Röhren, um Myonen nachzuweisen. Dabei ist das Myonspektrometer (siehe Abbildung 2.12) in vier Teildetektoren aufgebaut, von denen je zwei ausschließlich zur genauen Vermessung der Myonenimpulse dienen, während die anderen beiden zusätzlich noch schnelle ($\leq$ 25 ns) Triggerinformationen liefern. Insgesamt decken die vier Myonensysteme einen Pseudorapiditätsbereich von $|\eta| \leq 2.7$ ab, wobei die Triggerkammern auf einen Bereich $|\eta| \leq 2.4$ beschränkt sind (siehe auch Tabelle 2.3) und erreichen eine Impulsauflösung von $\sigma_{p_T}/p_T = 10\%$ bei $p_T = 1$ TeV.

2.2.3.1 Monitored Drift Tubes

Im Barrelbereich des Myonspektrometers kommen zur Präzisionsmessung der Myonenimpulse MDTs[27] zum Einsatz, die das hadronische Kalorimeter umgeben. Im Zentralbereich des Detektors werden dabei 3 zylindrisch angeordnete und parallel zur Strahlachse verlaufende Lagen von MDT-Kammern genutzt, während die Vorwärtsbereiche von vier ringförmigen Endkappen mit radial angeordneten Kammern abgedeckt werden. Die einzelnen MDT-Kammern bestehen aus 3 bis 8 Lagen von MDT-Röhren. Jede dieser Aluminiumröhren besitzt einen Durchmesser von etwa 3 cm und ist mit einer Ar/CO_2 Mischung gefüllt. Da die maximale Driftzeit innerhalb einer Röhre bis zu 700 ns betragen kann, können die MDTs nicht als Trigger eingesetzt werden. Mit einer Auflösung von bis zu 35 μm (bei Kombination hintereinander liegender Hits) sind die MDTs jedoch über weite Teile des Impulsraumes in der Lage, die Myonimpulse eigenständig zu bestimmen.

2.2.3.2 Cathode-Strip Chambers

Die Kathodenstreifenkammern CSC[28] kommen im Pseudorapiditätsbereich $2.0 \leq |\eta| \leq 2.7$ der inneren Myonenendkappe zum Einsatz, da hier auf Grund des hohen Teilchenflusses keine MDTs genutzt werden können. Etwa 7 m in Z vom Wechselwirkungspunkt entfernt, befinden sich daher je Detektorseite 16 trapezförmige CSC-Kammern mit jeweils 4 hintereinander liegenden CSC-Ebenen. Bei den CSCs handelt es sich im wesentlichen um Vieldraht-Proportionalkammern mit segmentierten Kathoden. Dabei sind die beiden gegenüberliegenden Platten einer CSC-Kammer mit um 90° versetzten Kathodenstreifen im Abstand von etwa 5 mm versehen, so dass eine ladungsgewichtete Auslese der Streifen eine 2D-Koordinateninformation liefert. Die CSCs sind ebenfalls mit einer Ar/CO_2 Gasmischung gefüllt und erzielen eine Auflösung von etwa 40 μm (für die zu den Drähten senkrechten Kathoden) in R und 5 mm in ϕ. Bedingt durch die hohe zeitliche Auflösung von unter 7 ns, kann das CSC Informationen über die korrekte Zuordnung von Events zu Beam-Crossings liefern.

[27] **M**onitored **D**rift **T**ube
[28] **C**athode **S**trip **C**hambers

2.2.3.3 Resistive Plate Chambers

Die RPCs[29] sind die im Zentralbereich des Myonspektrometers eingesetzten Triggerkammern. Dabei befinden sich je zwei RPC-Kammern auf jeder der mittleren Barrel-MDT-Kammern (eine auf der Innen-, eine auf der Außenseite) und je eine auf den äußeren MDT-Kammern (je nach Position, innen oder außen). Jede einzelne RPC-Kammer besteht dabei aus zwei "Einheiten", die wiederum aus einem Paar planparalleler, isolierender Plastikplatten bestehen. Durch Graphitelektroden wird ein etwa 10 kV starkes elektrisches Feld zwischen den etwa 2 mm voneinander entfernten Platten hergestellt. Durch ionisierenden Teilcheneinfall werden innerhalb des ebenfalls mit einem Ar/CO_2 Gasgemisch gefüllten Volumens Ladungslawinen erzeugt. Diese induzieren durch kapazitive Kopplung ein Signal auf den an der Außenseite der Plastikplatten angebrachten Auslesestreifen, die, wie im Fall der CSC, auf gegenüberliegenden Seiten um 90° verdreht sind. Die zwischen 23 mm und 35 mm voneinander entfernten Auslesestreifen ermöglichen dabei eine Ortsauflösung von etwa 10 mm sowohl in Z, als auch in ϕ. Die zeitliche Auflösung beträgt hingegen 1.5 ns, weshalb die RPC-Kammern auch Triggerinformationen liefern (siehe Abschnitt 2.2.4).

2.2.3.4 Thin Gap Chambers

Als TGCs[30] werden die im Vorwärtsbereich verwendeten Triggerkammern bezeichnet, die wiederum aus Vieldrahtproportionalkammern bestehen. Eine gegenüber den CSCs geänderte Kammer- und Auslesegeometrie, sowie die Verwendung eines speziellen $CO_2/n - C_5H_{12}$ Gasgemisches, machen diese Kammern sehr robust gegenüber der hohen Teilchenflussdichte im Vorwärtsbereich des Myonspektrometers und erlauben eine zeitliche Auflösung von $4ns$. Mit einer Ortsauflösung von etwa 7 mm in ϕ und 6 mm in R liefern die TGC-Kammern eine komplementäre Informationen zu den Koordinaten der MDT-Vorwärtskammern. Mechanisch sind die ebenfalls trapezoiden TGC-Einheiten an den inneren und mittleren Myonenendkappen unter Pseudorapiditäten von $1.92 \leq |\eta| \leq 2.4$ bzw. $1.05 \leq |\eta| \leq 1.92$ angebracht. Auf dem mittleren Myonenrad kommen dabei drei TGC-Einheiten mit insgesamt sieben TGC-Lagen zum Einsatz, während das kleine, innere Myonenrad nur eine zweilagige TGC-Einheit besitzt.

	Ausdehnung in η	**Hauptaufgabe**	**Genauigkeit**
MDT	$\|\eta\| \leq 2.7$ (innerste Lage: 2.0)	Tracking	35 μm(Z)
CSC	$2.0 \leq \|\eta\| \leq 2.7$	Tracking	40 μm(R)5 mm(ϕ)
RPC	$\|\eta\| \leq 1.05$	Triggering	10 mm(Z)10 mm(ϕ)
TGC	$1.05 \leq \|\eta\| \leq 2.7$	Triggering	$2-6$ mm(R)$3-7$ mm(ϕ)

Tabelle 2.3: *Hauptparameter der Subdetektoren des Myonsystems[3].*

[29] **R**esistive **P**late **C**hamber
[30] **T**hin **G**ap **C**hamber

2.2.4 Das Triggersystem

Eine zentrale Herausforderung im Betrieb des ATLAS-Detektors besteht in der Verarbeitung der anfallenden Daten. Ausgehend von einer Wechselwirkungsrate von $40 \cdot 10^9$ Hz, mit jeweils etwa 20 inelastischen Kollisionen, sowie einer Ereignisgröße von circa 1.3 MB ergibt sich eine Datenrate von mehr als $1 \cdot 10^{12}$ MB/s. Da diese Rohdatenmenge unmöglich vollständig und direkt gespeichert werden kann[31], ist es notwendig die Datenmenge vor der endgültigen Speicherung zu späteren Analysezwecken drastisch zu reduzieren. Dazu kommt im Falle des ATLAS-Detektors ein dreistufiges Triggersystem zum Einsatz, das die anfängliche Datenrate um den Faktor $4 \cdot 10^9$ auf etwa 250 MB/s reduziert, die auf Festspeichermedien geschrieben werden.

Das erste der drei Triggersysteme, der "Level 1"- oder kurze "L1"-Trigger ist dabei vollständig in programmierbarer Hardware implementiert und benötigt weniger als 2.5 μs für eine Entscheidungsfindung. Dabei berücksichtigt der L1-Trigger Informationen aus den Kalorimeter- und Myonensystemen, um Ereignisse mit hohem Transversalimpuls, hochenergetischen Elektronen, Photonen oder Jets, hadronisch zerfallenden τ-Leptonen oder hoher transversaler (oder auch fehlender) Energie zu selektieren. Um die geforderte hohe Selektionsrate von bis zu 75 kHz zu erreichen, nutzt der L1-Trigger eine gröbere Granularität der Kalorimeter, so dass je 16 der Kalorimetertower zu einem Triggertower zusammengefasst werden. Auf Grund der hohen benötigten Zeitauflösung werden für diesen Trigger weiterhin lediglich die schnellen RPC und TGC Myonenkammern genutzt, die zudem die Möglichkeit bieten, die betrachteten Ereignisse einem bestimmten Bunch-Crossing zuzuordnen. Neben der reinen Selektion interessanter Ereignisse, versieht der L1-Trigger beteiligte Detektorregionen ebenfalls mit einen RoI[32]-Flag, das von den nachfolgenden Triggersystemen genutzt wird. Durch den Einsatz programmierbarer Logikbausteine, lassen sich die Triggerbedingungen während des LHC-Betriebs anpassen, so dass beispielsweise die geforderte Energie oder Multiplizität eines bestimmten Triggerkanals (z.B. Elektronen) geändert werden kann. Während des L1-Entscheidungsprozesses von bis zu 2.5 μs werden die gesamten Rohdaten der einzelnen Subdetektoren entweder direkt im Detektor oder nahe am Detektor gepuffert, bevor die L1-Annahme im Mittel etwa eines von 10^7 Ereignissen, an die Auslesepuffer des Datennahmesystems weiterleitet.

Im Gegensatz zum hardwarebasierten L1 sind die beiden anderen ATLAS-Triggersysteme des HLT[33] durch spezielle Softwarealgorithmen, die auf einer Rechner-

[31] Um sich die Größe dieser Rohdatenmenge klarzumachen, ist eine einfache Beispielrechnung instruktiv: Geht man davon aus, dass weltweit 5 Milliarden Menschen je 1000 Gigabyte Datenmenge pro Jahr erzeugen, so beläuft sich die daraus resultierende jährliche Gesamtdatenrate der Menschheit auf $5 \cdot 10^{15}$ MB/a. Allein die Rohdaten des ATLAS-Experiments hingegen würden sich auf mehr als 10^{12} MB/s $\cdot\, 3 \cdot 10^7$ s/a $\approx 5 \times 6000 \cdot 10^{15}$ MB/a summieren. Damit wäre die Rohdatenmenge des ATLAS-Experiments pro Stunde etwa so groß, wie die von der gesamten Menschheit pro Jahr produzierte Datenmenge!

[32] **R**egion **o**f **I**nterest

[33] **H**igh **L**evel **T**rigger

Farm ausgeführt werden, realisiert. Insgesamt kommen dabei mehr als 1000 Rechner und etwa 10000 CPUs zum Einsatz, deren genaue Aufgabenteilung auf die beiden "Level 2"- und "Event Filter"-Triggersysteme den Anforderungen des laufenden Betriebs angepasst werden kann.

Der L2-Trigger hat die Aufgabe, aus den eingehenden 75 kHz L1-Annahmen etwa 3 kHz für den nachfolgenden Event Filter auszuwählen. Dazu nutzt der L2-Trigger die vom L1-Trigger bereitgestellten RoI-Informationen, um gezielt zusätzliche Daten (beispielsweise Daten höherer Granularität im Falle der Kalorimeter) aus den RoI-markierten Detektorbereichen anzufordern. Je nach Konfiguration werden etwa 25% der verfügbaren HLT-Rechnerknoten für den L2 eingesetzt, woraus eine durchschnittliche Bearbeitungszeit von etwa 40 ms pro Knoten und Event resultiert. Neben der Selektion der relevanten Ereignisse werden am Ende des L2-Entscheidungsprozesses erstmals vollständige Ereignisse aus den Detektordaten rekonstruiert, die dann an den nachfolgenden Event Filter übergeben werden.

Der Event Filter setzt sich ebenfalls aus einer Rechnerfarm zusammen, deren Aufgabe die Filterung der vom L2-Trigger ausgewählten und bereits teilweise rekonstruierten Ereignisse besteht. Dazu werden sowohl die L1-RoI- als auch die zusätzlich vom L2-Trigger hinzugefügten Ereignisinformationen genutzt. Um die endgültige Datenrate von etwa 200 Hz zu erreichen, muss die L2-Ausgangsrate von etwa 3000 Hz nochmals reduziert werden. Der Event Filter kann dabei auf die im Vergleich mit den anderen Triggersystemen aufwändigsten Algorithmen zurückgreifen, wobei pro Entscheidungsprozess und CPU etwa 4 s zur Verfügung stehen.

Nach der abschließenden Filterung durch den Event Filter werden die Daten mit einer maximalen Datenrate von 400 Hz (durchschnittlich 200 Hz) zunächst auf ein lokales RAID[34]-System geschrieben von wo aus sie gepuffert und in verschiedene Physikkanäle eingeteilt auf das zentrale CERN-Speichersystem übertragen werden.

[34] **R**edundant **A**rray of **I**ndependent **D**isks

Kapitel 3

Der ATLAS-Pixeldetektor

Der in Abschnitt 2.2.1.1 bereits eingeführte Pixeldetektor ist der innerste Subdetektor des ATLAS-Experiments. Als Teil des inneren Spurdetektors liefert der Pixeldetektor dabei wichtige Information für die Rekonstruktion der Flugbahnen geladener Teilchen und damit zur Bestimmung ihrer Impulse und Ladungsvorzeichen. Neben dieser, ebenfalls von den SCT- und TRT-Detektoren ausgeführten Aufgabe (siehe Abschnitt 2.2.1.2 und 2.2.1.3), ist der Pixeldetektor weiterhin für die Rekonstruktion primärer sowie sekundärer Vertizes verantwortlich.

Die intrinsische Unsicherheit bei der Bestimmung der Ortskoordinaten des primären Vertex ist dabei durch die Größe des Überlappungsvolumens zweier Bunches, innerhalb dessen inelastische Wechselwirkungen stattfinden können, sowie deren Position relativ zum Koordinatenursprung gegeben.

Rekonstruierbare sekundäre Vertizes entstehen durch den Zerfall langlebiger instabiler Teilchen, wie B-Mesonen oder Kaonen (K_S^0).

Die genaue Bestimmung der Teilchenimpulse und die Rekonstruktion der Vertizes stellen hohe Anforderungen an das räumliche Auflösungsvermögen des Pixeldetektors. Hinzu kommen weitere Rahmenbedingungen in Form des hohen Teilchenflusses und der, damit direkt zusammenhängend, eingeforderten Strahlenhärte der Detektorkomponenten (siehe Abbildungen 2.7 und 2.8). Da sich der Pixeldetektor zwischen den Vertizes und allen anderen Subdetektoren des ATLAS-Experiments, insbesondere den Kalorimetern, befindet, lag ein wesentlicher Gesichtspunkt bei der Entwicklung des Detektordesigns auf der Minimierung der durch den Pixeldetektor verursachten Strahlungs-, sowie Wechselwirkungslängen (siehe dazu auch Abbildung 3.1).

3.1 Detektoraufbau

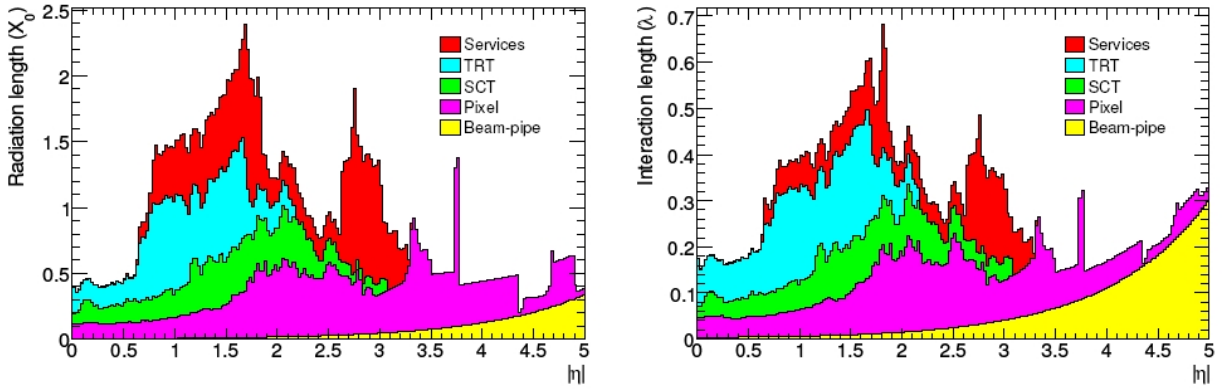


Abbildung 3.1: *Anteil der einzelnen Subdetektoren an der Wechselwirkungslänge, bzw. der Strahlungslänge in Abhängigkeit von η[3].*

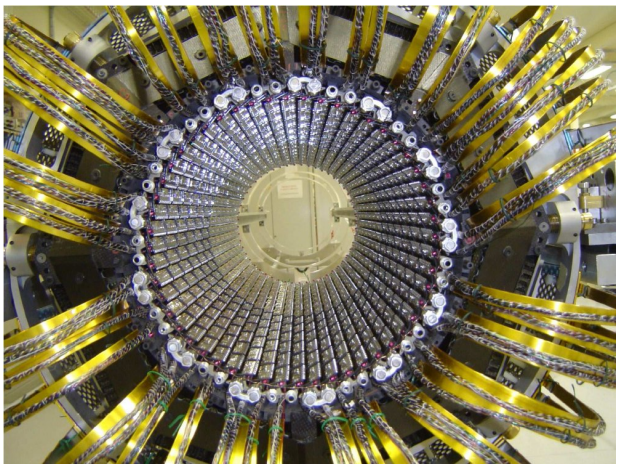


Abbildung 3.3: *Fotografie des "Layer-2" des Pixeldetektors[3].*

Wie alle ATLAS-Subdetektoren verfügt auch der Pixeldetektor über eine Tonnen- bzw. Zwiebelschalengeometrie, um einen Pseudorapiditätsbereich von $|\eta| \leq 2.5$ abzudecken. Dabei kommen im Zentralbereich drei zylindrische und zur Strahlachse konzentrische Lagen, B-Layer (oder auch Layer-0), Layer-1 und Layer-2 zum Einsatz. Mit einem Radius von lediglich 50.5 mm besitzt der B-Layer den geringsten Abstand von den Wechselwirkungspunkten, weshalb ihm bezüglich der Rekonstruktion sekundärer Vertizes eine besondere, namensgebende Bedeutung zukommt. Die beiden Lagen Layer-1 und Layer-2 folgen dabei mit Radien von 88.5 mm und 122.5 mm (jeweils mittlere, effektive Modulradien) auf die innerste Lage. Alle drei Lagen besitzen dieselbe "aktive" (also mit Sensoren bestückte) Ausdehnung von ± 400 mm in Z.

Um den oben angegebenen Pseudorapiditätsbereich zu überdecken, kommen im Vorwärtsbereich sechs ebenfalls zur Strahlachse konzentrische Scheiben oder "Disks" zum Einsatz. Je drei der identischen Disks, mit Innen- und Außenradien von 88.8 mm bzw. 149.6 mm, wurden dabei auf der A- und der C-Seite des Detektors in Abständen von ± 495 mm, ± 580 mm und ± 650 mm angebracht.

Die Grundbausteine des Pixeldetektors sind die 1744 (bis auf die Anschlussverkabelung) identischen Detektormodule, die in Abschnitt 3.2 genauer beschrieben werden.

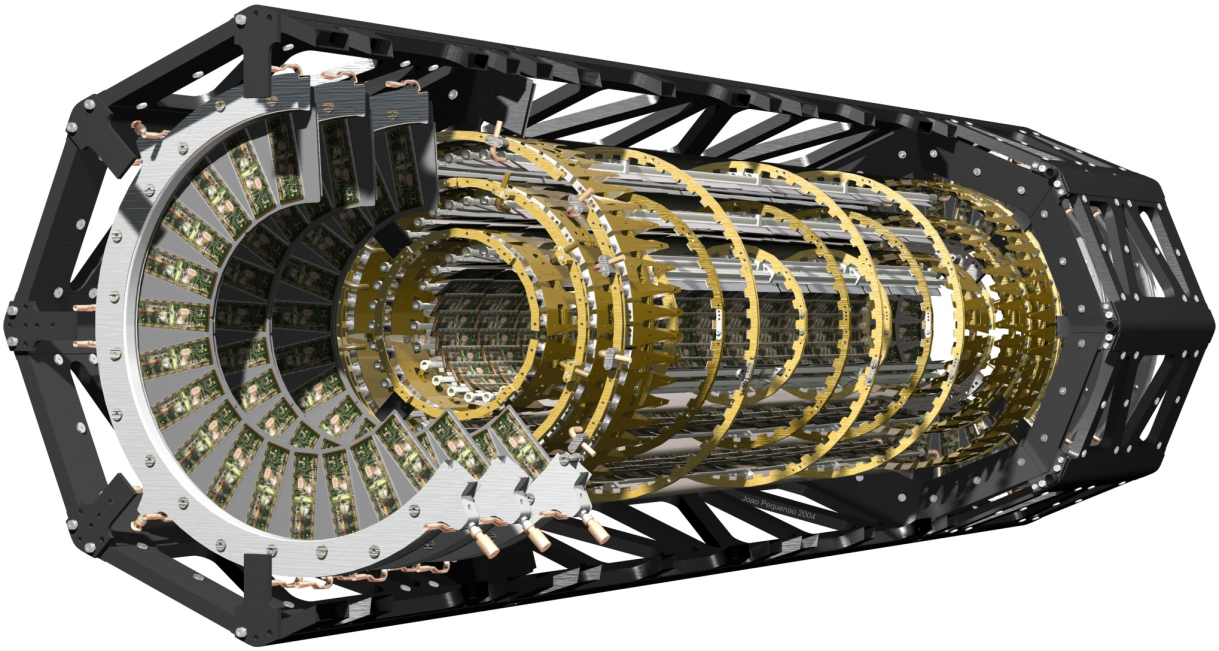


Abbildung 3.2: *Risszeichnung des ATLAS-Pixeldetektors[3].*

3.1.1 Zentralbereich

Im durch die Zylinderlagen abgedeckten "Barrelbereich" des Pixeldetektors sind die Module auf etwa 16 mm breiten und 830 mm langen "Staves" angeordnet. Jeder Stave verläuft dabei parallel zur Strahlachse und trägt 13 Detektormodule, die dem Wechselwirkungspunkt entgegengeneigt sind. Dabei ist lediglich das zentrale Modul, seine Bezeichnung lautet "M0", planparallel zum Stave ausgerichtet. Alle anderen Module, also je sechs auf der A- und auf der C-Seite, neigen sich um 1.1° dem mittleren Modul und damit dem Wechselwirkungspunkt entgegen, was eine in Z überlappende Konstruktion der Modulenden auf einem Stave erlaubt. Die sechs Module der A-Seite werden von der Stavemitte (oder "M0") ausgehend mit "M1A" bis "M6A" bezeichnet, wobei die Nomenklatur auf C-Seite analog "M1C" bis "M6C" lautet.

Die Staves selbst sind pro Lage durch zwei "Half-Shells" (also Halbschalen) fixiert, die die zylindrische Struktur der Lagen definieren. Um auch in ϕ eine überlappende Anordnung der Module zu gewährleisten, werden die einzelnen Staves um etwa 20° um ihre Längsachse rotiert installiert. Diese Konfiguration hat weiterhin den Vorteil, dass es damit möglich ist, den durch Lorentzdrift der Signalelektronen im $\vec{E} \times \vec{B}$-Feld hervorgerufenen Lorentzwinkel zu kompensieren und die Ladungsverteilung auf einzelne Sensorzellen zu optimieren (siehe "Ladungsgewichtung", Abschnitt 3.2).

Da die Detektormodule zur Reduzierung der zu erwartenden Strahlenschäden bei einer Temperatur von -10 °C betrieben werden müssen, befindet sich auf der dem Strahlrohr abgewandten Seite eines jeden Staves ein Kühlröhrchen. Die einzelnen Röhrchen werden dabei von C_3F_8 durchströmt, dass unmittelbar vor den Röhrchen spezielle, Druck mindernde Kapillare durchläuft und dann in den Kühlröhrchen verdampft. Dies führt zu einer hohen Wärmeaufnahme des Kühlmittels und einer damit einhergehenden Kühlung der Module. Gleichzeitig wird damit der Anteil des Kühlsystems an der Gesamtstrahlungslänge im Vergleich zu einer Einphasenkühlung verringert, da das Kühlmittel bei gleicher Wärmetransportfähigkeit eine geringere Dichte aufweist.

Je zwei Staves werden durch ein U-förmiges Verbindungsstück hintereinandergeschaltet und bilden einen vom Rest des Pixeldetektors unabhängig zu betreibenden Kühlkreislauf[1]. Ein solches, durch einen gemeinsamen Kühlkreislauf gekoppeltes Stave-Paar wird als "Bi-Stave" bezeichnet.

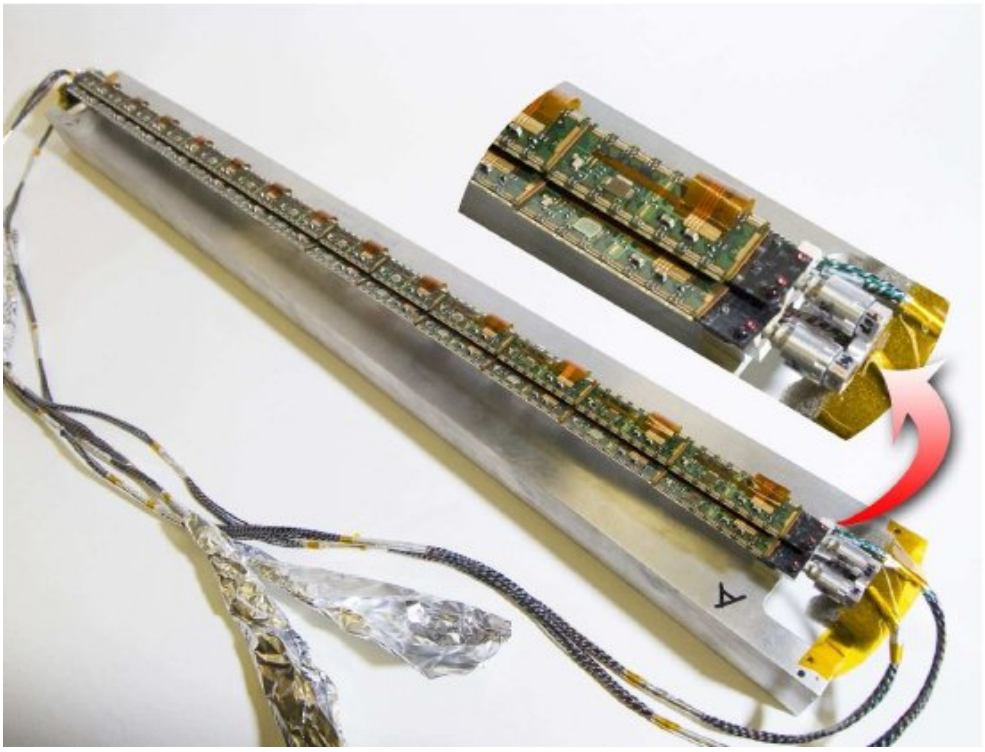


Abbildung 3.4: *Fotomontage eines Bi-Staves. Im vergrößerten Endbereich sind die Anschlüsse für die Verdampfungskühlung zu erkennen[3].*

Je nach Durchmesser besitzen die drei Zylinderlagen des Pixeldetektors 11, 19 oder 26 dieser Bi-Staves mit den Bezeichnungen "B01-B11", "B01-B19" und "B01-B26". Die wichtige Rolle der Kühlkreisläufe (auch PCC[2] genannt) wird spätestens bei der Namensgebung

[1] Auf nicht direkt am Detektor befindlichen Komponenten dieses Kühlsystems, wie etwa Kondensatoren, Kompressoren oder Druckregulatoren, soll hier nicht weiter eingegangen werden.

[2] **P**arallel **C**ooling **C**ircuit

der Staves deutlich, die nicht etwa pro Lage durchnummeriert werden, sondern immer nur in Verbindung mit der Benennung ihres Bi-Staves mit “Bxx_S1” oder “Bxx_S2” bezeichnet werden.

Bedingt durch die Granularität der optischen Auslese, sowie einiger Spannungsversorgungen, die im weiteren noch genauer erläutert werden sollen, gibt es im Barrelbereich noch eine weitere erwähnenswerte Hierarchieebene zwischen den Staves und den Modulen: die “Half-Staves”. Wie der Name vermuten lässt, besteht ein Stave immer aus zwei Half-Staves, wobei die ungerade Anzahl an Stavemodulen dazu führt, dass einer dieser Half-Staves sechs und der andere sieben Module besitzt. Dabei ist die Aufteilung so durchgeführt, dass der erste vom Kühlmittel durchflossene Stave “S1_A6” auf A-Seite sechs Module hat und der benachbarte Half-Stave “S2_A7” sieben (das mittlere Modul wird also aus Symmetriegründen einmal zur A- und einmal zur C-Seite des Detektors gezählt.)

Insgesamt besteht der Zentralbereich des Pixeldetektor aus 56 Bi-Staves, 112 Staves, 224 Half-Staves und somit 1456 Detektormodulen mit einer Gesamtsensorfläche von etwa 1.45 m^2.

3.1.2 Vorwärtsbereich

Auf den Disks des Vorwärtsbereichs werden die Module nicht wie auf den Staves, parallel zur Strahlachse, sondern radial dazu auf “Sektoren” angeordnet. Jede Disc besteht aus 8 Sektoren die wiederum jeweils 6 Module tragen, die mit “M1” bis “M6” bezeichnet werden.

Da im Gegensatz zum SCT, die Module des Pixeldetektor überall identisch und damit rechteckig (und nicht etwa trapezförmig) sind, werden pro Sektor 3 Module auf jeder Seite versetzt angebracht, um die ansonsten entstehenden Sensorlücken zu schließen. Abbildung 3.5 zeigt ein Foto eines solchen Disk-Sektors. Bedingt durch die Parallelität von $\vec{E}$- und $\vec{B}$-Feldern im Vorwärtsbereich des Pixeldetektors ist hier allerdings keine Korrektur auf Grund einer etwaigen Lorentzdrift notwendig.

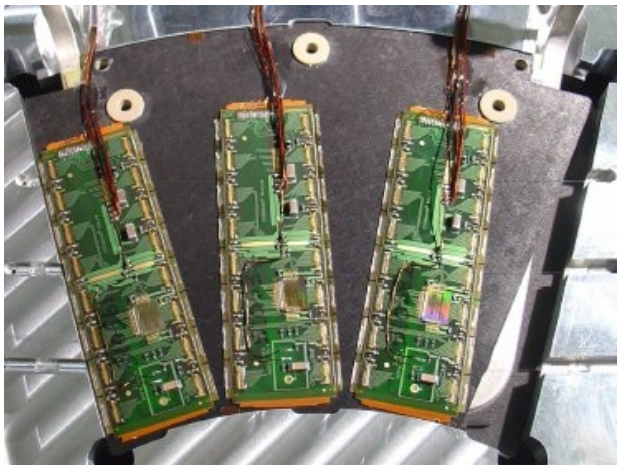


Abbildung 3.5: *Fotografie eines Endkappen-Sektors. Die drei rückwärtigen Module sind auf dem Bild nicht zu erkennen. Diese sind so angeordnet, dass die keilförmigen Lücken zwischen den Modulen der Vorderseite von den Modulen der Rückseite überdeckt werden[3].*

Da die Anzahl der Module pro Sektor im Endkappenbereich bereits der Auslesemodularität entspricht, ist die Einführung einer weiteren Hierarchieebene hier nicht nötig. Die Sektoren entsprechen somit mechanisch den Staves und elektrisch den Half-Staves.

Ähnlich der Anordnung im Barrelbereich des Detektors, werden auch in den Endkappen je zwei Sektoren "S1" und "S2" über ein Verbindungsstück zu einem autarken Kühlkreislauf mit der Bezeichnung "Bi-Sector" (oder ebenfalls PCC) verbunden. Jede der sechs Discs "D1A" bis "D3C" besteht aus vier dieser Kühlkreisläufe "B01" bis "B04", so dass die Endkappensektion insgesamt $6 \cdot 4 \cdot 2 \cdot 6 = 288$ Module mit einer Sensorfläche von etwa 0.3 m^2 aufweist.

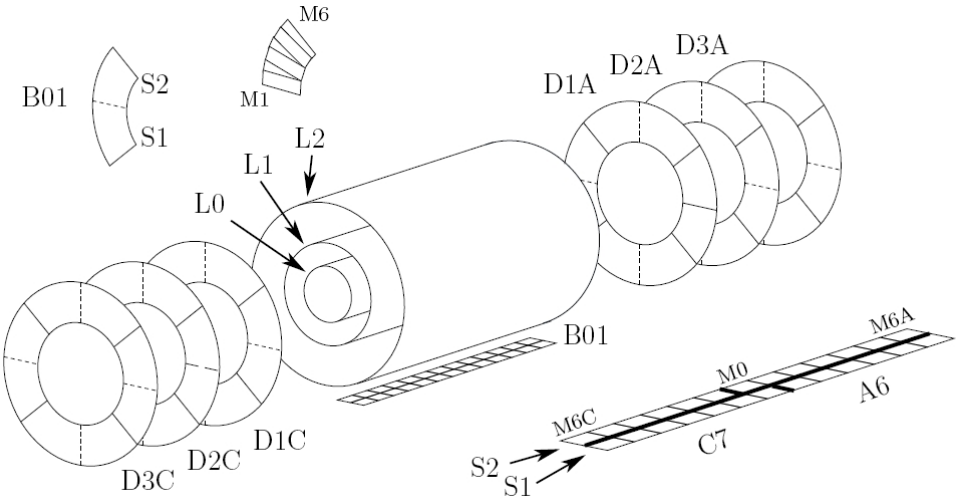


Abbildung 3.6: *Veranschaulichung der Namenskonvention zur Bezeichnung der Module innerhalb des Detektors[41].*

Die Gesamtheit aus Disks und Barrellagen wird von einer oktagonalen Trägerstruktur mit einer Länge von 1442 mm und einem Maximaldurchmesser von 430 mm präzise (bis auf etwa 50 μm) in Position gehalten. Die Anforderungen an die Trägerstruktur des Pixeldetektors sind hoch. Die Trägerstruktur muss nicht nur hochpräzise gefertigt sein, sondern auch möglichst kleine thermische Ausdehnung aufweisen und möglichst wenig zur Gesamtstrahlungs-/Gesamtwechselwirkungslänge des Pixeldetektors beitragen. Um dies zu erreichen, bestehen alle wesentlichen Trägerstrukturen des Detektors aus Kohlefaserverbundwerkstoffen.

Pixeldetektor-Namensschema

Während es mit den Modulseriennummern und den in der Datennahme verwendeten "Hash-IDs" weitere Namensschemata für die Bezeichnung der Pixelmodule gibt, ist das für das Detektorkontrollsystem und damit für diese Arbeit relevante Namensschema der Detektorgeographie folgend hierarchisch aufgebaut.

Der Name eines Detektormoduls wird aus den beteiligten Hierarchiestufen zusammengesetzt, wobei diese zusätzlich durch "_" voneinander getrennt werden.

Beispiele für Modulnamen nach diesem oft auch "geographisch" genannten Namensschema sind "**L0_B07_S2_C7_M2A**" oder "**D2C_B03_S1_M6**".

3.2 Das Pixelmodul

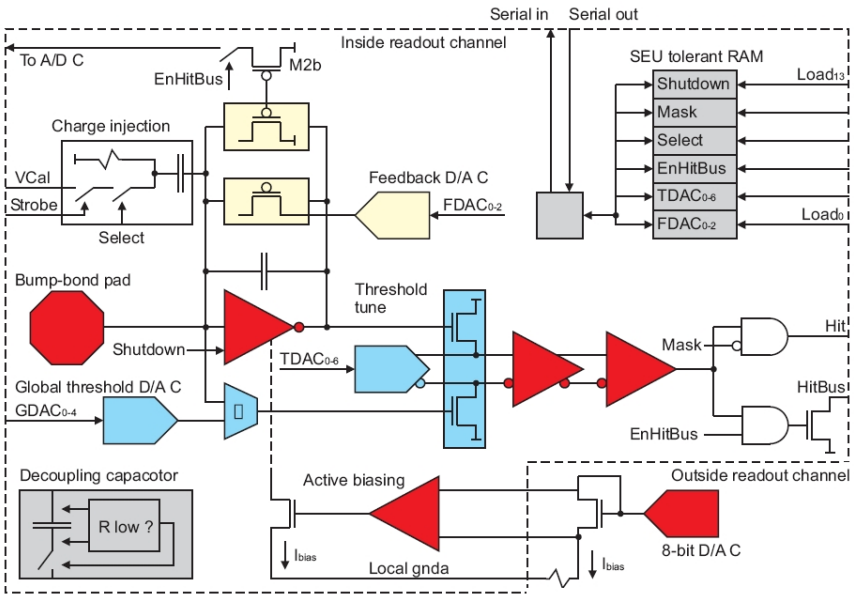


Abbildung 3.7: *Auslesezelle eines einzelnen Pixelsensors[10].*

Jedes der 1744 Module des Pixeldetektors besitzt 46080 Pixelsensoren, von denen die meisten die Dimension 400 μm $\times$ 50 μm bei einer Sensordicke von 250 μm besitzen[3]. Je 2880 der Sensorzellen werden von einem der 16 FE[4]-Chips eines Moduls abgedeckt, der

[3]Um ein hermetisches Detektordesign auch zwischen den Auslesechips zu ermöglichen kommen hier spezielle, 600 μm $\times$ 50 μm lange Pixelzellen zum Einsatz.

[4]**F**ront **E**nd

mit Hilfe von kleinen Lötkügelchen, "Bump-Bondings" genannt, mit den einzelnen Zellen verbunden ist. Der FE-Chip besitzt pro Zelle eine eigene Ausleseelektronik, die in 3.7 abgebildet ist.

Die im aktiven Sensorvolumen des Pixels erzeugten Elektronen laden über die Bump-Bonding-Verbindung einen Kondensator auf, dessen Entladestrom mit einem 3-Bit FDAC[5] konfiguriert werden kann. Ein Verstärker (in Abbildung 3.7 rechts neben dem "Bumbbond pad" eingezeichnet) leitet das Signal an einen Diskriminator weiter, der das Signal mit Hilfe einer einstellbaren Schwelle digitalisiert. Dabei lässt sich diese über einen 5-Bit GDAC[6] global pro FE-Chip, sowie über einen 7-Bit TDAC[7] lokal pro Pixel einstellen.

Neben der eigentlichen digitalen Trefferinformation erzeugt die Elektronik noch Zeitstempel für den Zeitpunkt des Überschreitens der Diskriminatorschwelle während des Ladevorgangs, sowie das Unterschreiten der Schwelle beim Entladevorgang. Die Differenz der beiden Werte in Einheiten des LHC-Taktes, die "TOT[8]", ist auf Grund der linearen Lade- und Entladeströme ein Maß für die durch Ionisation in der Pixelzelle freigesetzte Ladung.

Bedingt durch die Tatsache, dass die TOT-Werte einer Sensorzelle mit der in der Zelle deponierten Ladung zusammenhängen, ist es möglich, die Präzision der räumlichen Auflösung bei Treffern, die in mehreren benachbarten Pixelzellen registriert wurden, durch Ladungsgewichtung zu erhöhen. Weiterhin bietet die Sensorzellenelektronik die Möglichkeit, eine definierte Probeladung in die Elektronik zu speisen, sowie mehrere Schalter zum Maskieren oder kompletten Deaktivieren der Sensorzelle. Die Konfiguration der damit insgesamt 14-Bit breiten Zellenregister erfolgt über ein serielles Interface durch den FE-Chip. Die 16 FE-Chips eines Moduls werden wiederum von einem gemeinsamen MCC[9] angesteuert und ausgelesen. Der MCC selbst ist dabei über LVDS[10]-Leitungen mit der Auslesekette des Pixeldetektors verbunden.

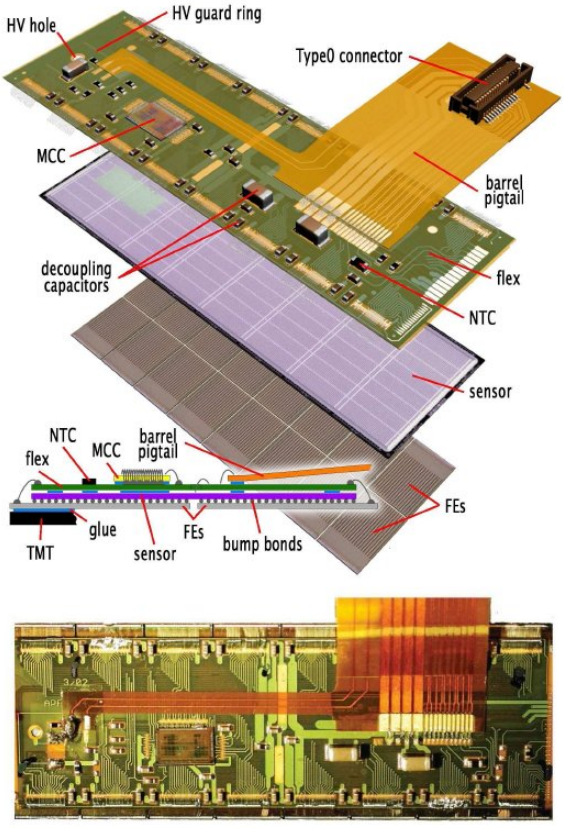


Abbildung 3.8: *Oben: Explosionszeichnung und Schema eines Pixelmoduls (nähere Beschreibung siehe Text) Unten: Fotografie eines Moduls. Die gezeigte Aufsicht entspricht der dem Wechselwirkungspunkt zugewandten Seite. Beide Abbildungen zeigen ein Barrelmodul mit dem charakteristischen Pig-Tail[3].*

[5] **F**eedback **D**igital **A**nalog **C**onverter
[6] **G**lobal **DAC**
[7] **T**hreshold **DAC**
[8] **T**ime **O**ver **T**hreshold
[9] **M**odule **C**ontrol **C**hip
[10] **L**ow **V**oltage **D**ifferential **S**ignal

Abbildung 3.8 zeigt eine schematische Darstellung eine Pixelmoduls. Die einzelnen Frontend-Chips sind über "Wire-Bonds" mit einer flexiblen Kaptonfolie mit aufgedruckter Schaltung, kurz Flex-Platine genannt verbunden. Auf dieser befinden sich neben dem MCC auch einige Kondensatoren, sowie ein NTC[11]-Temperatursensor zur Bestimmung der Modultemperatur. Der einzige Unterschied zwischen den Disk- und Barrelmodulen des Pixeldetektors besteht im Anschluss der Flexplatine an die Auslesekette. Im Falle der radial angebrachten Discmodule können die notwendigen "Microwires" direkt mit dem stirnseitigen Flex verbunden werden, während auf den Barrelmodulen ein "Pig-Tail" zum Einsatz kommt, der aus einer dünnen Kaptonfolie besteht, mit deren Hilfe die notwendigen Leiterbahnen auf der dem Modul abgewandten Seite eines Staves kontaktiert werden können.

Da die Frontend-Chips die aktive Sensorfläche nicht vollständig überdecken, kommen in den Randbereichen zwischen den FE-Chips spezielle Pixelzellen mit geänderten Dimensionen, bzw. geänderter Ausleseansteuerung zum Einsatz, auf die hier jedoch nicht näher eingegangen werden soll (eine detaillierte Beschreibung findet sich beispielsweise in [10]).

3.2.1 Die optische Datenübertragung

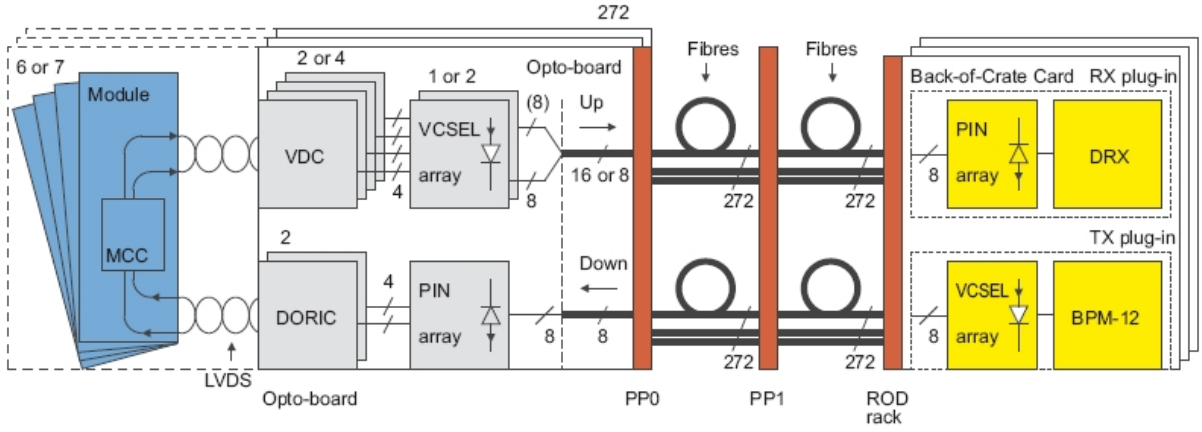


Abbildung 3.9: *Der optische Link des Pixeldetektors und seine Komponenten[3].*

Bedingt durch die hohen Anforderungen an Datenrate, Strahlentoleranz und Materialeinsparung wird für die Datenübertragung zwischen Pixeldetektor und Ausleseraum ein optischer Kommunikationspfad genutzt. Da sowohl auf den Auslesecrates als auch auf den Detektormodulen elektrische Signale verwendet werden, sind auf beiden Seiten des optischen Links entsprechende Wandler notwendig, die die optischen Signale erzeugen bzw. empfangen können.

[11] **N**egative **T**emperature **C**oefficient

Auf Modulseite kommen "Optoboards" zum Einsatz, welche über etwa 1 m lange Kabel mit den Modulen elektrisch verbunden sind. Diese befinden sich am Patch Panel 0 (siehe auch Abbildung 4.2), am Ende der äußeren Supportstrukturen, die den Pixeldetektor an beiden Endkappen mit den Versorgungsleitungen verbinden.

Jedes der insgesamt 272 Optoboards ist für einen Half-Stave oder Sector, also je 6 oder 7 Detektormodule, zuständig. Die Aufgabe der Optoboards liegt dabei in der Vermittlung und Umwandlung der elektrischen Signale der MCCs bzw. der optischen Steuersignale des Ausleseraums.

Für das Senden optischer Lichtimpulse nutzen die Optoboards je 8 VCSEL[12]. Der VDC[13] empfängt dabei die Daten der an das Optoboard angeschlossenen MCCs und erzeugt daraus elektrische Steuersignale für die VCSE-Laser. Da, bedingt durch die unterschiedliche durchschnittliche Okkupanz der Module, die Datenrate für B-Layer Optoboards deutlich höher ausfällt als für die anderen Optoboards des Pixeldetektors, sind diese mit je zwei 8-Kanal VCSEL-Arrays ausgestattet, was die Datenübertragungsrate von 80 Mb/s auf 160 Mb/s verdoppelt[14].

Um die optischen Datensignale der im Ausleseraum befindlichen Datennahmegeräte entgegen zu nehmen, besitzt jedes Optoboard des weiteren ein 8-Kanal PIN[15]-Diodenarray, dessen Signale mit Hilfe des DORIC[16] in das zur Ansteuerung der MCCs benötigte elektrische Format konvertiert werden.

Auf der dem Detektor abgewandten Seite des optischen Links kommen die in Auslesecrates installierten BOCs[17] zum Einsatz. Diese verfügen ebenfalls über VCSEL-Arrays für die Übertragung der Daten an die Optoboards, sowie über PIN-Diodenarrays für den Empfang der Detektordaten. Je nach Datenrate (40 Mb/s für Layer-2, 160 Mb/s für B-Layer und 80 Mb/s sonst) können ein, zwei oder vier Optoboards an einen BOC angeschlossen werden. Analog zu den Optoboards befinden sich auf den BOCs die notwendigen Bauteile, um die VCSEL und PIN-Dioden anzusprechen. Die BOCs fungieren dabei als elektro-optischer Wandler der ROD[18]-Karten, die Teil der ATLAS-weiten DAQ-Hardware sind..

Je eine Glasfaserleitung pro Modul überträgt Clock- und Steuersignale von den Ausleseeinheiten zu den Detektormodulen, während für die Rückrichtung, den eigentlichen Datentransfer der Module, je nach Lage ein oder zwei (für den B-Layer) Fasern genutzt werden.

[12] **V**ertical **C**avity **S**urface **E**mitting **L**aser
[13] **V**CSEL **D**river **C**hip
[14] Die benötigte Datenrate für Blayer, Layer-1/Disks und Layer-2 sind 160 Mb/s, 80 Mb/s und 40 Mb/s.
[15] **P**ositive **I**ntrinsic **N**egativ
[16] **D**igital **O**pto **R**eceiver **I**ntegrated **C**ircuit
[17] **B**ack **o**f **C**rate Card
[18] **R**ead **O**ut **D**river

Kapitel 4

Das Detektorkontrollsystem (DCS)

4.1 Der Aufbau des DCS

Das Detektorkontrollsystem des Pixeldetektors ist für die Überwachung und Kontrolle der Spannungen, Ströme und Temperaturen der einzelnen Detektormodule, sowie der zugehörigen Infrastruktur wie beispielsweise Optoboards oder Spannungsregulatoren zuständig. Um diese Aufgabe zu erfüllen, besteht das DCS aus einer ganzen Reihe verschiedener Hardwarekomponenten. Diese müssen dabei nicht nur korrekt mit den restlichen Komponenten verkabelt, sondern auch mittels spezieller Softwarepakete in die gemeinsame Kontrollsoftware eingebunden werden.

Um einen Überblick über das gesamte DCS zu geben, bietet es sich an, den Signal-, bzw. Kontrollweg der einzelnen Untersysteme wie beispielsweise Niederspannungs- oder Hochspannungssystem getrennt zu betrachten.

Die vorliegende Beschreibung orientiert sich an Abbildung 4.1. Hier sind die verschiedenen Hardwarekomponenten des DCS in Abhängigkeit von der Entfernung vom Wechselwirkungspunkt innerhalb des ATLAS-Detektors dargestellt. DCS-Komponenten befinden sich grundsätzlich an den verschiedenen Umverteilungsstationen der zahlreichen Kabelstränge (PP[1], siehe Abbildung 4.2). Einige der Komponenten haben daher auch Namen, die die grobe Position verraten (z.B. HV-PP4).

Fast alle vorgestellten Hardwarekomponenten kommunizieren durch den Einsatz von ELMBs[2] mit den DCS-Rechnern, auf denen die Kontrollsoftware läuft. Dabei kommt ausschließlich Microsofts OPC[3] zum Einsatz. Bei Komponenten, die keine ELMB für die

[1] **P**atch **P**anel
[2] **E**mbedded **L**ocal **M**onitoring **B**oard
[3] **O**bject linking and embedding for **P**rocess **C**ontrol

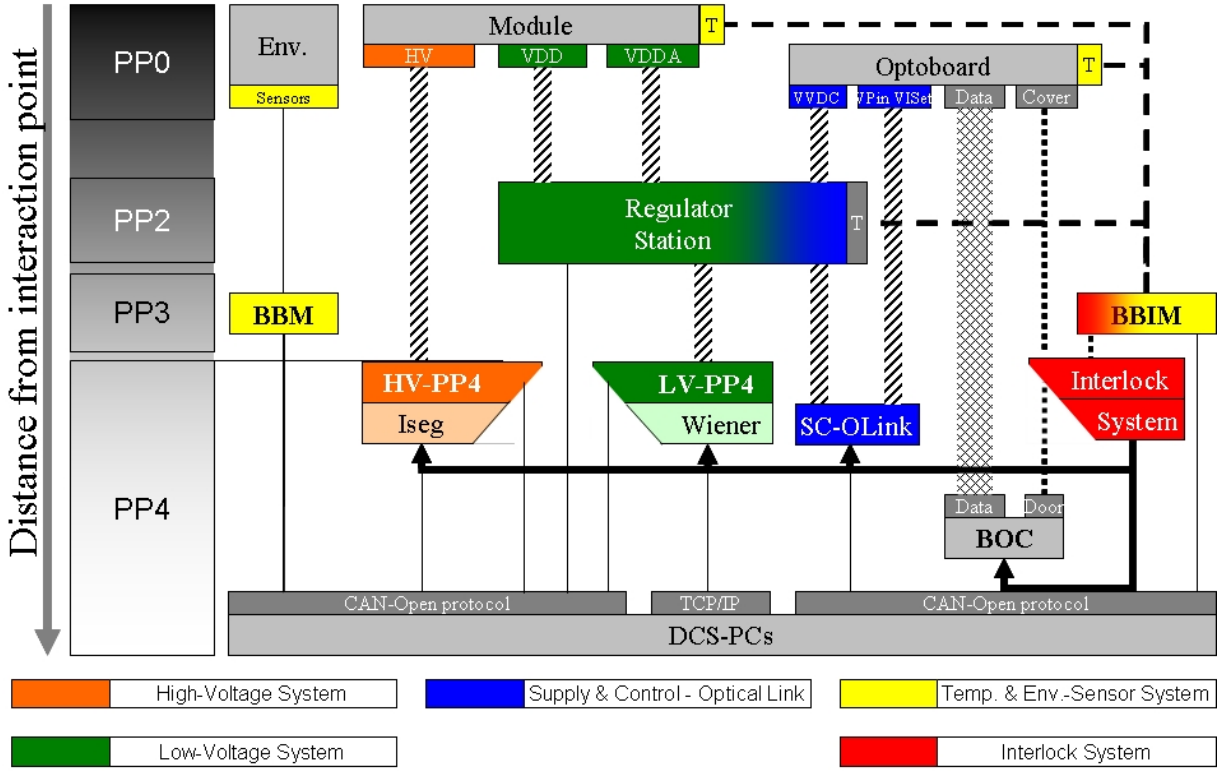


Abbildung 4.1: *Prinzipieller Aufbau der DCS-Hardware.*

Auslese der Daten nutzen, wird darauf gesondert hingewiesen. Der verwendete Feldbus ist in den meisten Fällen CAN[4], wobei auch hier bei Abweichung darauf hingewiesen wird.

4.1.1 Das Hochspannungssystem

Für den Betrieb der einzelnen Sensorzellen wird eine variable Depletionsspannung von bis zu 700 V benötigt, um erhöhten Spannungsbedarf verursacht durch Strahlungsschäden im Halbleitermaterial ausgleichen zu können. Die eigentliche HV-Versorgungsspannung wird dabei von Hochspannungsmodulen der Firma iseg[5] geliefert. Da die Modularität, also die Anzahl der angeschlossenen Detektormodule pro Hochspannungskanal eines iseg-Moduls, zunächst 6 bzw. 7 und später 2 sein wird, müssen besondere Vorkehrungen hinsichtlich der Anbindung der Module an die Hochspannungsquellen getroffen werden.

Die Änderung der Modularität wird notwendig, da ein einzelner iseg-Kanal bei 700 V nur bis zu 4 mA Strom liefern kann und damit nicht genug Leistung für den störungsfreien Betrieb von mehr als 2 Detektormodulen bei maximaler Depletionsspannung liefert. Auf

[4] **C**ontroller **A**rea **N**etwork
[5] iseg Spezialelektronik GmbH, Radeberg, Deutschland, früher: **I**nstitut für **S**pezial**e**lektronik **G**mbH

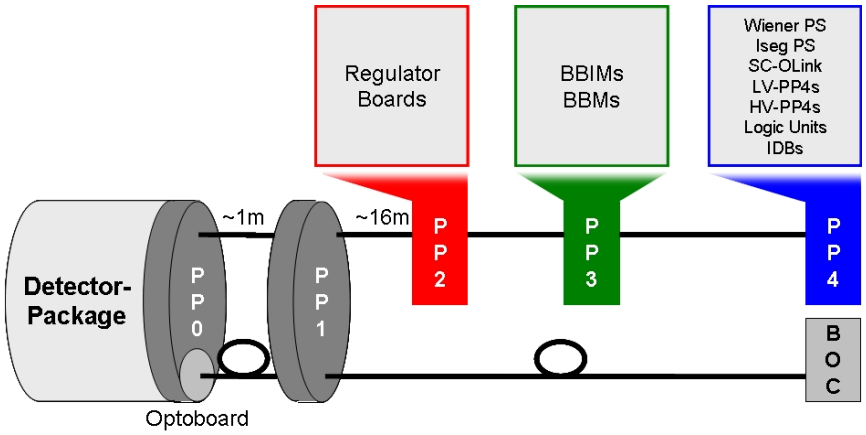


Abbildung 4.2: *Zuordnung der DCS-Hardwarekomponenten zu den einzelnen Patch-Paneln [37][modifiziert]*

Parameter	Spannung	Strom	Temperatur
HV	700 V	≤ 2 mA	
VDD	2.1 V	700 mA	
VDDA	1.7 V	1200 mA	
Modultemperatur			-10 °C
VISet	0.9 V	0.4 mA bis 2 mA	
VPin	20 V	20 mA	
VVDC	10 V	800 mA	
Rst_opto	2.5 V		
Optoboardtemperatur			20 °C

Tabelle 4.1: *DCS-Betriebsparameter der Detektormodule und Optoboards während der Datennahme.*

Grund des hohen Preises der Hochspannungsmodule werden diese zusätzlichen Module allerdings erst bei Bedarf nachgerüstet.

Zwischen den iseg-Spannungsversorgungen und den Detektormodulen auf Höhe des PP4 (Ausleseraum US15 bzw. USA15) befinden sich daher zusätzliche, HV-PP4 genannte HV-Patchpanel, die für die Umverteilung der einzelnen Kanäle, sowie die genaue Strommessung jedes einzelnen Detektormoduls zuständig sind und die die erwähnte Umstellung der Modularität ermöglichen.

Die iseg-Hochspannungsmodule kommunizieren direkt über CAN mit den, auf den DCS-Rechnern installierten, OPC-Servern (ein Server stellt die Daten der mit ihm kommunizierenden Hardware beliebig vielen Benutzern zur Verfügung. Siehe Abschnitt 4.2).

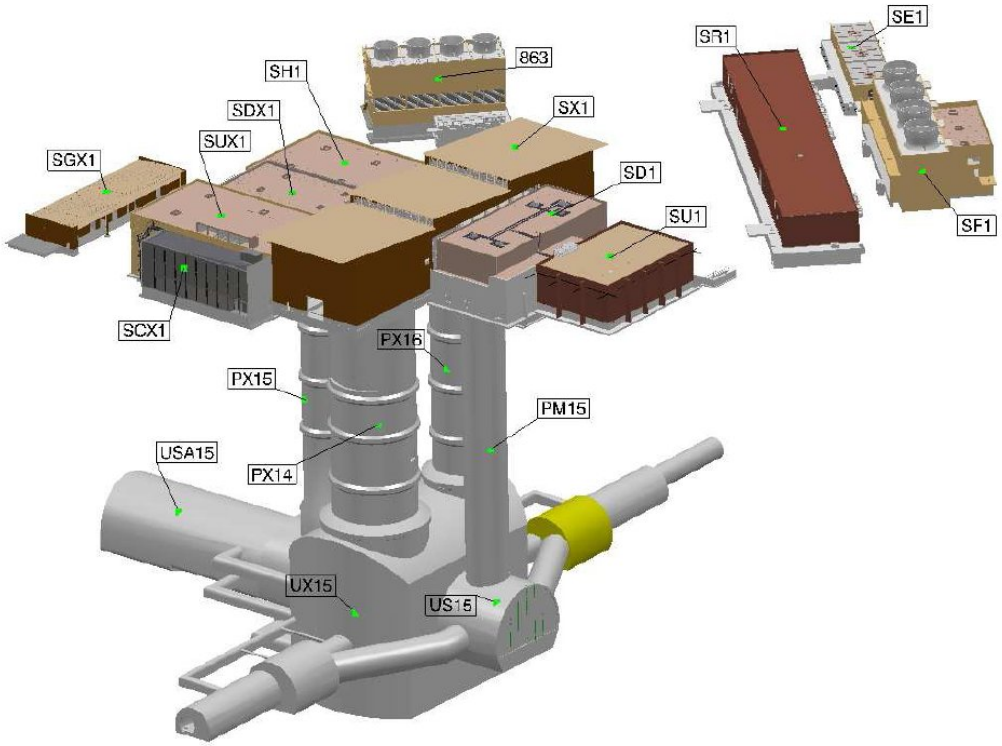


Abbildung 4.3: *Darstellung der ATLAS-Kaverne und des zugehörigen Gebäudekomplexes. Für DCS relevant sind neben der eigentlichen ATLAS-Kaverne vor allem die beiden Ausleseräume "US15" und "USA15", der ebenerdig gelegene ATLAS-Kontrollraum "SCX1" sowie die Testhalle "SR1", in der sich das auch während des LHC-Betriebs weitergenutzte Pixel-Testsystem befindet[3].*

4.1.2 Das Niederspannungssystem

Im Gegensatz zum Hochspannungssystem kann die Niederspannung für die Versorgung der analogen und digitalen Modulelektronik nicht direkt von den Niederspannungsversorgungen, die sich aus Gründen der Strahlentoleranz und des Platzbedarfs auf Höhe des PP4 befinden, geliefert werden. Der Grund hierfür liegt in den hohen Spannungsabfällen in den bis zu 100 m langen Niederspannungskabeln (offensichtlicherweise ist dieser Abfall für die Hochspannungsversorgung zu vernachlässigen).

Andererseits wäre der Einsatz der Versorgungscrates nahe am Detektor nur möglich, wenn hier spezielle, strahlentolerante oder gar strahlenharte Hardware zum Einsatz käme. Das von der Pixel-Kollaboration gewählte Szenario sieht daher den Einsatz herkömmlicher Spannungsversorgungen (Off-the-Shelf) in Kombination mit speziellen detektornahen und strahlenharten Spannungsregulatoren vor.

Während die Niederspannungsversorgungen der Firma W-Ie-Ne-R[6] sich genau wie die Hochspannungsversorgungen im strahlengeschützten Ausleseraum "US15" bzw. "USA15" (siehe Abbildung 4.3) bei PP4 befinden, werden "Regulator-Stations" auf Höhe des PP2 installiert (innerhalb des ATLAS-Detektors, aber außerhalb des Inneren-Detektors und der Kalorimeter).

Jede dieser Regulator-Stations beherbergt bis zu 12 Regulator-Boards, deren Aufgabe die direkte Regulierung der von den W-Ie-Ne-R-Power-Supplies gelieferten Spannungen an die Detektormodule ist. Die unvermeidlichen Spannungsabfälle zwischen Ausleseraum, Regulator Stations und Detektormodulen werden dazu über Sense-Wires auf den Modulen gemessen und entsprechend kompensiert. Ein einzelnes Regulator-Board bedient dabei immer einen Half-Stave oder Sector (also bis zu 7 Detektormodule) mit analoger und digitaler Versorgungsspannung (VDDA bzw. VDD). Dabei wird jedes Regulator-Board jeweils von einem W-Ie-Ne-R-Kanal mit der notwendigen Spannung für VDD, sowie von einem weiteren Kanal mit VDDA versorgt, so dass pro Half-Stave / Sector zwei W-Ie-Ne-R-Kanäle sowie ein Regulator-Board benötigt werden.

Um die W-Ie-Ne-R-Niederspannungen auf die einzelnen Regulator-Kanäle zu verteilen und weiterhin eine Möglichkeit zu haben, die einzelnen VDD- bzw. VDDA-Ströme der Module getrennt voneinander zu bestimmen, kommt, ähnlich dem HV-PP4, zudem ein LV-PP4 genanntes Auslesecrate zum Einsatz. Wie der Name schon andeutet, befinden sich diese Crates ebenfalls im Zählraum, in unmittelbarer Nähe der W-Ie-Ne-R-Power-Supplies (Entfernung: wenige 10 cm).

Im Gegensatz zu allen anderen Hardwarekomponenten nutzen die W-Ie-Ne-R-Crates nicht den CAN-Feldbus, sondern das TCP/IP[7]-Protokoll, sowie einen eigenen OPC-Server. Technisch gesehen ist das Verfahren allerdings transparent (d.h. die Kontrollsoftware bemerkt keinen Unterschied) und erfordert lediglich den Einsatz eines privaten Netzwerks, das innerhalb der ATLAS-Kaverne allerdings vorhanden ist.

4.1.3 Das Versorgungssystem für den optischen Link

Der bereits angesprochene optische Link besteht aus den im Counting-Room befindlichen BOCs, den detektorseitigen und bereits erwähnten Optoboards, sowie den für den Betrieb der Optoboards notwendigen Versorgungseinheiten, den SC-OLinks[8].

Letztere versorgen die Optoboards mit jeweils drei Spannungen „VVDC", „VPin" und „VISet". VPin und VISet, die die PIN-Dioden versorgen und den VCSEL-Strom einstellen, können auf Grund der geringen Last direkt von den im Counting-Room installierten SC-OLinks versorgt werden. Im Gegensatz dazu muss die VVDC-Versorgungsspannung

[6] **W**erk für **I**ndustrie**e**lektronik - **N**uklear**e**lektronik - **R**egelungstechnik, Plein & Baus GmbH, Burscheid, Deutschland

[7] **T**ransmission **C**ontrol **P**rotocol / **I**nternet **P**rotocol

[8] **S**upply and **C**ontrol for the **O**pto**l**ink

des VDCs innerhalb des Regulator-Boards auf Grund der deutlich höheren Ströme und dadurch bedingten Spannungsabfälle reguliert werden.

Für jedes der 272 Optoboards wird von den SC-OLink-Karten ferner ein Resetsignal bereitgestellt, mit dem der optische Link zurückgesetzt werden kann. Insgesamt fallen also $272 \cdot 4$ zu kontrollierende Kanäle für den DCS-Teil des optischen Links an (ohne BOC-Parameter).

Ein SC-OLink-Crate besteht dabei aus bis zu 16 SC-OLink-Karten mit jeweils einem komplexen Kanal (ein Kanal führt hierbei jeweils VVDC, VPin, VISet und Reset).

4.1.4 Das Temperatur-Überwachungssystem

Das hohe Strahlungsniveau im Bereich des Pixeldetektors führt zu vorzeitiger, strahlenbedingter Alterung der Sensorzellen. Dabei kommt es zu verschiedenen Effekten, die im Laufe der Betriebsdauer zu einer notwendigen Erhöhung der Depletionsspannung führen (anfangs führt ein Ladungsträgerinversion genannter Prozess hingegen dazu, dass die Depletionsspannung für eine Weile gesenkt werden kann). Die komplexen Strahlungsschädigungen der Kristallstruktur des Siliziumsensors hängen von mehreren Faktoren ab, von denen die Temperatur einer der wichtigsten ist. Eine niedrige Temperatur führt dabei zu einem geringeren Ausmaß der Strahlenschäden.

Daher wird der gesamte Pixeldetektor während des Experiments bei einer Temperatur von -10 °C betrieben.

Da ein Temperaturanstieg negative Auswirkungen für die einzelnen Module hat und bei hohen Temperaturen sogar zu direkten thermischen Schäden der Elektronik führen kann, wird die Temperatur jedes einzelnen Moduls von DCS überwacht. Die von den Modul-NTCs gelieferten Temperatur-Spannungen werden zu den BBIMs[9] geführt, die mit Hilfe eingebauter Interlock-Boxen mittels einer Komparatorschaltung digitale Interlocksignale aus den analogen Spannungswerten erzeugen. Die analogen Spannungen werden parallel von einer ELMB gelesen und an die Steuerungssoftware des DCS weitergegeben.

Neben den Modultemperaturen sind auch die Temperaturen der Optoboards, der On-Detector-Komponenten des optischen Links, von großer Wichtigkeit. Um ein Überhitzen der Laser- und PIN-Dioden zu verhindern, müssen auch die Optoboardtemperaturen ständig überwacht werden.

Komplettiert wird das Temperatur-Überwachungssystem durch die Temperaturen der eingesetzten Regulator-Boards, deren Überwachung ebenfalls von DCS übernommen wird.

[9] **B**uilding **B**lock **I**nterlock **M**onitoring

4.1.5 Das Interlocksystem

Die Aufgabe des "Interlocksystems" ist es, den Pixeldetektor vor potentiellen Schäden zu schützen und die Sicherheit des Betriebspersonals im Zusammenhang mit den eingesetzten infraroten Lasern des optischen Links zu gewährleisten. Um im Falle einer kritischen Parameteränderung (beispielsweise eines raschen Temperaturanstiegs) schnell und zuverlässig reagieren zu können, werden alle kritischen Betriebsparameter vom Interlocksystem überwacht. Das Interlocksystem ist dabei unabhängig von der für die restliche Systemkontrolle eingesetzten Software und bietet mit dem Einsatz schneller FPGAs[10] eine zuverlässige und redundante Kontrollmöglichkeit.

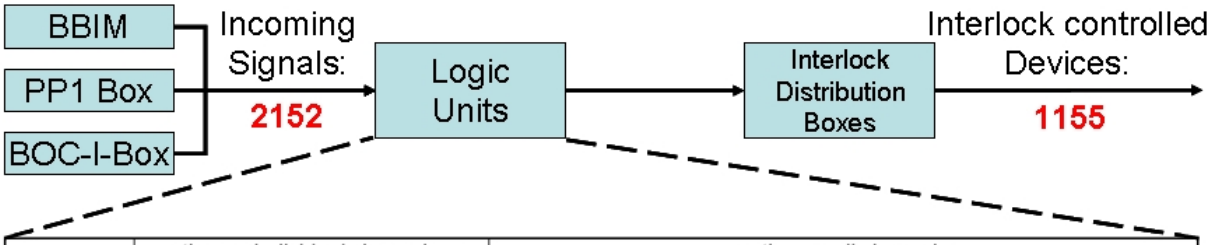
Das Interlocksystem besteht aus den bereits erwähnten und für die Digitalisierung der kritischen Temperaturen zuständigen BBIMs, sowie weiteren Komponenten. Um die Sicherheit des Betriebspersonals zu gewährleisten, sind auf Detektorseite und im Counting-Room, auf beiden Seiten des optischen Links, Schließkontakte angebracht (Gefahr für das menschliche Auge durch die intensive, infrarote Laserstrahlung). Sollte einer der Kontakte geöffnet werden, wird sofort der gesamte optische Link deaktiviert. Die dafür zuständigen Interlock-Crates heißen auf Counting-Room-Seite „BOC-I-Box“ und auf Detektorseite „I-PP1-Box“.

Neben den bisher erwähnten Interlocksignalen, die von den Sensoren „T_Module“, „T_Regulator“, „T_Optoboard“, „I_BocDoor“ sowie "I_OptoboardCover" generiert werden, gibt es drei weitere Pixel-externe Interlocksignale „I_DSS[11]“. I_DSS0 ist ein ATLAS-globales Interlocksignal, das im Falle eines generellen Problems mit dem Kühlsystems, eines Feuers oder anderweitiger schwerwiegender Probleme ausgelöst wird. I_DSS1 und I_DSS2 sind speziellere Signale, die durch den Strahlzustand (I_DSS1) bzw. durch Zustand des Pixel- und SCT-Kühlsystems generiert werden. Der I_DSS2-Interlock ist insbesondere im Zusammenhang mit dem "Bake out" des Strahlrohrs wichtig, da im Fall eines Ausfalls des Pixeldetektor-Kühlsystems sofort die Strahlrohr-Heizung ausgeschaltet werden muss.

Alle diese Signale werden in den "Logic Units" entgegengenommen und mit Hilfe von FPGAs ausgewertet. Die Logic Units selbst erzeugen dabei logische Signale, die dann an die IDBs[12] weitergeleitet werden (siehe Abbildung 4.4). Dort wird mit Hilfe weiterer FPGAs rekonstruiert, welche Kanäle welcher Spannungsversorgungen "geinterlockt" werden müssen (das "Interlocken" ist dabei vom einfachen Abschalten eines Kanals zu unterscheiden, da die Hersteller immer einen separaten Interlockeingang pro Kanal bieten, der auch per Software ausgelesen werden kann).

Obwohl betont werden sollte, dass das Interlocksystem vollkommen unabhängig von der DCS-Software operiert, so ist doch zu erwähnen, dass dies ein passives Monitoring

[10] **F**ield **P**rogrammable textbfGate **A**rray
[11] **D**etector **S**afety **S**ystem
[12] **I**nterlock **D**istribution **B**ox



	action on individual channels or small groups of channels			action on all channels					
	HV	LV	SC-OLink	Bakeout Sys.	Opto heater	HV	LV & PP2 aux. PS	SC-OLink	BOC
T_{Module}	X	X		X (T_{Module} of Blayer)					
$T_{Regulator}$		X	X						
$T_{Optoboard}$			X		X				
$I_{BocDoor}$			X						X (Crates of one rack)
$I_{OptoboardCover}$									X
I_{DSS0}					X	X	X	X	
I_{DSS1}						X			
I_{DSS2}				X					

Abbildung 4.4: *Schematische Übersicht des Interlock-Systems und der Interlock-Matrix [16][7]*

des Interlocksystems durch die DCS-Software nicht ausschließt. Ganz im Gegenteil ist ein solches Monitoring sehr hilfreich, wenn es gilt, die tatsächlich gesetzten Interlocks mit den Zuständen der Power-Supplies zu vergleichen und die Ursache eines Interlocksignals zurückzuverfolgen. Als DCS-relevante Parameter werden daher alle Ein- und Ausgänge des Interlocksystems, sowie die zur Verfügung gestellten Testsignale[13] von DCS überwacht.

4.1.6 Das Überwachungssystem für die Umgebungssensoren

Neben der Vielzahl der bereits erwähnten Spannungs- und Temperatursensoren werden weitere Sensoren die DCS-Datennahme komplettieren, die nicht direkt einer bestimmten Detektorkomponente zugeordnet sind. Beispiele für solche Sensoren sind zusätzliche NTCs oder die für die Bestimmung des Taupunktes wichtigen Feuchtigkeitssensoren.

Alle diese Sensoren werden unter dem Stichwort „Environmental Sensors“ zusammengefasst und von den BBM[14]-Crates ausgelesen. Da diese Sensoren keine direkt kritischen

[13] mit denen sich Interlocks simulieren lassen
[14] **B**uilding **B**lock **M**onitoring

Messdaten liefern, wird hier auf die Digitalisierung zwecks Einspeisung ins Interlocksystem verzichtet (daher der Unterschied zum BBIM).

4.1.7 Infrastruktur-Systeme

Die in den vorangegangenen Abschnitten vorgestellten Hauptsysteme des Pixel-DCS tragen zu einer für ein Prozessleitsystem sehr hohen Anzahl von Prozessparametern bei, die in Abbildung 4.5 dargestellt sind. Der Vollständigkeit halber seien hier noch die bisher nicht eingeführten aber in der Tabelle berücksichtigten Teilsysteme erwähnt. Dazu gehören einige W-Ie-Ne-R-VMEbus[15]-Crates, die beispielsweise für die Spannungsversorgung der über VMEbus angeschlossenen BOC- und ROD-Crates zuständig sind. Von besonderer Wichtigkeit ist dabei die Tatsache, dass der VMEbus dieser BOC- und ROD-Crates über die von DCS angesteuerten W-Ie-Ne-R-VMEbus-Crates resettet werden kann. Weiterhin werden pro BOC je 4 Temperatursensoren vom DCS ausgelesen. Neben diesen, zum Betrieb der DAQ-Hardware notwendigen Systemen, kommen "Auxilliary"-Power-Supplies für die Spannungsversorgung der BBIM-Crates, die CAN-Bussysteme und die PP2-Crate-Controller zum Einsatz, die ebenfalls über DCS überwacht und angesteuert werden.

Während der Testphase des Pixeldetektors wurde festgestellt, dass einige der 272 eingesetzten Optoboards bei niedrigen Temperaturen ein zu langsames Ansteigen des VCSEL-Lichtleistung zeigten, das sich bei den während der Qualifikationsphase bei Raumtemperatur betriebenen Optoboards nicht gezeigt hatte. Um diesem als "Slow-Turn-On" bezeichneten Effekt entgegenzuwirken, wurde ein separates Heizsystem installiert, das den Betrieb der Optoboards in einem vom restlichen Pixeldetektor-Kühlsystem unabhängigen und für die Optoboards idealen Temperaturbereich (≥ 20 °C) erlaubt.

[15] **V**ersa **M**odule **E**urocard **bus** (Datenbussystem mit bis zu 64 Bit Busbreite).

System	Subsystem	Devices	# of Channels	Geo. Granularity	Parameter	# of Params/Ch	Total
Low-Voltage	**VDD**	Wiener	272	Readout Units	On/Off, Vmeas, Imeas, Vset, Iset	5	**1360**
		Regulator Boards	1744	Modules	On/Off, Vmeas, Vset	3	**5232**
		LV_PP4s	1744	Modules	Imeas	1	**1744**
	VDDA	Wiener	272	Readout Units	On/Off, Vmeas, Imeas, Vset, Iset	5	**1360**
		Regulator Boards	1744	Modules	On/Off, Vmeas, Vset	3	**5232**
		LV_PP4s	1744	Modules	Imeas	1	**1744**
High-Voltage	**HV**	Iseg	272	Readout Units	On/Off, Vmeas, Imeas, Vset, Iset	5	**1360**
		HV_PP4s	1744	Modules	Imeas	1	**1744**
Optical Link	**Vpin**	SC-Olink	272	Readout Units	On/Off, Vmeas, Imeas, Vset	4	**1088**
	VISet	SC-Olink	272	Readout Units	On/Off, Vmeas, Imeas, Vset	4	**1088**
	VVDC	SC-Olink	272	Readout Units	On/Off, Vmeas, Imeas, Vset	4	**1088**
		Regulator Boards	272	Readout Units	On/Off, Vmeas, Vset	3	**816**
	Reset Signal	SC-Olink	272	Readout Units	On/Off	1	**272**
Temp. & Env. Sensors	**Tmodule**	BBIMs	1744	Modules	Temperature	1	**1744**
	Topto	BBIMs	272	Readout Units	Temperature	1	**272**
	Env. Sensors	BBMs	400			1	**400**
Interlock System	**interlock in (temp)**	Logic Units	28	detector quarters	interlock input signals		**2512**
	interlock out	IDBs	16	detector quarters	interlock output signals		**1088**
	Bakeout System Interlock	BOB	2	detector	interlock in- and output signals		**167**
	OptoHeater Interlock	OH-Ibox	2	detector	interlock in- and output signals	15	**30**
	Laser Interlock + general safety	PP1 boxes	2	detector	interlock signals		**16**
DAQ Hardware	**VME crates**		10	crate	2 x temp, 3x (Vmon, Imon), ON/Off, reset	10	**100**
	Tboc		132	BOC	temperature	4	**528**
Opto heaters			48		Tset, Tmon, On/Off, power	4	**192**
auxiliary power supplies	**PP2 aux**		56		On/Off, Vmeas, Imeas, Vset, Iset	5	**280**
	PP3 aux		14		Vmon, Imon	2	**28**
	CAN PS		29		2 * (Vmon, Imon), On/Off	5	**145**
Pixel DCS			**13362**				**31630**

Abbildung 4.5: *Prozessparameter des Pixeldetektor-Kontrollsystems.*

4.2 Die DCS-Software

Da das DCS mehrere 10000 Betriebsparameter überwachen und kontrollieren muss (siehe Abbildung 4.5), ist eine Automatisierung mit Hilfe geeigneter Software unabdingbar. Mindestanforderung der Automatisierungsbestrebungen ist dabei, das spätere Schichtpersonal ohne die zwingende Anwesenheit eines Experten in die Lage zu versetzen, das DCS zu bedienen, und die Sicherheit nicht zu gefährden.

4.2.1 PVSS

Da einerseits viele der gegebenen Problemstellungen im Bereich des DCS für alle LHC-Detektoren sehr ähnlich ausfallen und andererseits ein leichterer Austausch von Expertenwissen gewährleistet werden sollte, entschied sich das zuständige Kontrollgremium im Vorlauf der LHC-Experimente für den LHC-weiten Einsatz von PVSS[16].

Das kommerzielle PVSS-System der Firma ETM[17] bietet dabei eine hohe Anpassungsfähigkeit und Flexibilität. Ohne hier auf die genauen Details der Software eingehen zu wollen, sei gesagt, dass es sich bei PVSS im Grunde um eine integrierte Entwicklungsumgebung (also ein Softwarepaket aus Editor, Compiler/Interpreter, sowie graphischer Darstellungsmöglichkeit) für Prozessleitsysteme handelt.

Wesentliches Augenmerk wurde bei der Entwicklung von PVSS auf die Ausfallsicherheit des Systems gelegt, was zu teilweise ungewöhnlichen aber sehr stabilen Konzepten führte. Hierbei ist zu beachten, dass Daten innerhalb von PVSS in "Datenpunkten" dargestellt werden. Diese Datenpunkte ähneln in vielerlei Hinsicht den Klassen in C++. Insbesondere können Datenpunkte eine beliebige, hierarchische Struktur aus verschiedenen Datenpunktelementen aufweisen.

Im Gegensatz zu Variablen in vielen anderen Programmiersprachen werden die Datenpunkte in PVSS nicht im Arbeitsspeicher, sondern direkt auf der Festplatte aufbewahrt. Zwar zieht dies einen Geschwindigkeitsnachteil nach sich, jedoch führt es dazu, dass alle Daten bei einem Systemabsturz gesichert sind.

Ein weiteres wichtige Merkmal der Software ist die Aufteilung der Software in mehrere "Manager" genannte Rechner-Prozesse. Jeder dieser Manager übernimmt dabei spezifische Teilaufgaben, so dass es beispielsweise Userinterface-, Archiv- oder Treibermanager gibt. Untereinander kommunizieren diese mit Hilfe von TCP/IP, wobei ein "Event"-Manager die Zusammenarbeit der anderen Manager steuert. Alle zu einem Event-Manager gehörenden Manager bilden ein "PVSS-Projekt", oder mit anderen Worten eine autark lauffähige PVSS-Instanz. Mehrere dieser Projekte können über "Distribution"-Manager miteinander verbunden werden, so dass ein Prozessleitsystem aus mehreren Teilsystemen bestehen

[16] **P**rozess **V**isualisierungs und **S**teuerungs **S**oftware, ETM AG, Eisenstadt, Österreich
[17] **E**lektro**t**echnik **M**ühlgassner

kann, die unabhängig voneinander funktionsfähig bleiben, jedoch zentral gesteuert werden können. Durch den Einsatz des Netzwerkprotokolls TCP/IP sowohl für die Kommunikation zwischen Manager als auch für die Verbindung von Projekten besitzt ein PVSS-Prozessleitsystem einen hohen Grad an Flexibilität.

Durch den von PVSS unterstützten Einsatz von OPC fällt die aufwändige Programmierung von speziellen Gerätetreibern großteils weg und der Entwickler kann sich auf die eigentlich zu implementierende Funktionalität der Software konzentrieren.

Die Kommunikation zwischen einem beliebigen Gerät und der Software erfolgt dabei (zumindest in unserem Fall) ausschließlich über den OPC-Standard (PVSS unterstützt auch andere Protokolle). Bei dieser Art der Kommunikation stellt der Gerätehersteller einen OPC-Server zur Verfügung, der direkt mit Geräten des entsprechenden Typs kommuniziert. Weitere Applikationen (wie beispielsweise PVSS) können dann über eigene OPC-Clients Verbindung mit dem Server aufnehmen. Der große Vorteil dieses Systems liegt darin, dass man bei **N** Geräten und **M** Applikationen nicht **N**·**M** Treiber sondern nur **N** Server und **M** Clients zur Verfügung stellen muss (**N**+**M**).

PVSS besitzt also nicht **M** verschiedene OPC-Clients, sondern lediglich einen. Dieser Universalclient verfügt über alle notwendigen Fähigkeiten, um mit jeder beliebigen OPC-Hardware zu kommunizieren.

In einem abstrakten Beispiel sieht das so aus, dass ein Gerät seine Betriebsparameter an den OPC-Server meldet. Dort werden die Parameter OPC-Items genannt. Diese lassen sich mit einem beliebigen OPC-Client lesen/schreiben (natürlich sind nicht alle Items bidirektional). Der OPC-Client von PVSS kann die OPC-Items dann in das PVSS-System übertragen.

Üblicherweise stellt der Entwickler Skripte (also komplexere Funktionen) zur Verfügung, mit deren Hilfe auf PVSS-Seite alles automatisiert wird, was zur Anbindung eines neuen Gerätes notwendig ist:

- Erstellung der notwendigen Datenpunkte
- Modifikation der OPC-Server-Konfigurationsdatei (denn diese erzeugt die OPC-Items)
- Neustart von OPC-Server und Client
- Verbindung von OPC-Items und Datenpunktelementen

Eine weitere Möglichkeit, Daten innerhalb von PVSS zu verarbeiten, ist die Verbindung zweier (oder mehrerer) Datenpunktelemente mittels “Datenpunktfunktionen”. Diese speziellen Funktionen schaffen eine permanente unidirektionale Verbindung zwischen einem oder mehreren Quelldatenpunktelementen und einem Zieldatenpunktelement. Ein gutes

Beispiel für eine Datenpunktfunktion ist die Berechnung einer Temperatur aus der gemessenen NTC-Spannung und der Referenzspannung des Sensors.

Um die parallele Arbeit mehrerer Entwickler an der Software zu erleichtern, bietet PVSS das Konzept der Subprojekte, bei dem ein PVSS-Programm (in der PVSS-Sprache ein Projekt) aus mehreren Subprojekten bestehen kann. Die Pixel-DCS-Gruppe hat von diesem Konzept von Anfang an Gebrauch gemacht, so dass sich die verschiedenen Softwarekomponenten strikt voneinander getrennt in verschiedenen Subprojekten befinden.

Zum jetzigen Zeitpunkt kann man die Subprojekte jedoch in drei unterschiedliche Klassen einteilen, die im Folgenden kurz eingeführt werden sollen. Eine Übersichtsabbildung über die Hauptsoftwarekomponenten findet sich in Abbildung 4.6.

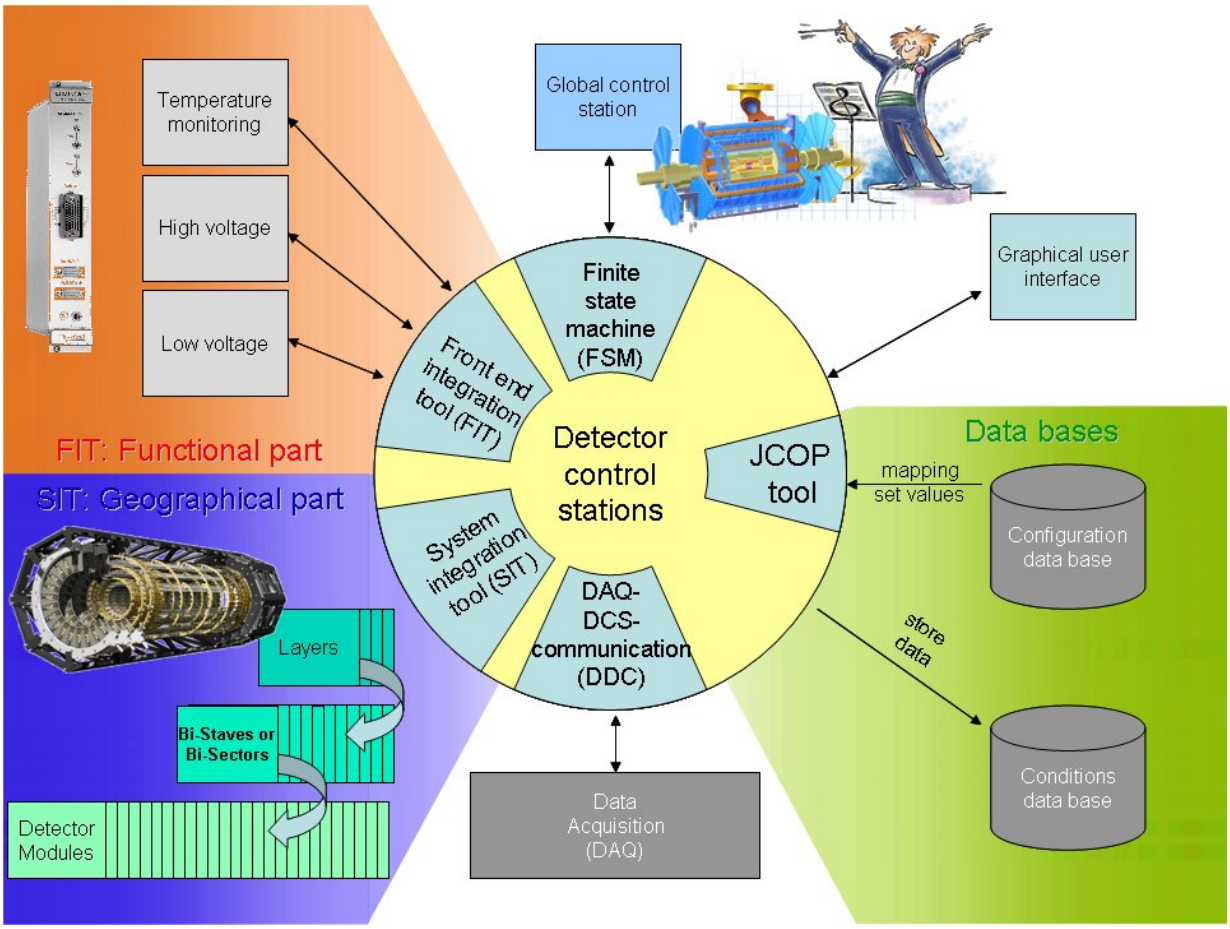


Abbildung 4.6: *Übersicht der Pixel-DCS-Software*

4.2.2 Die Front-End Integration Tools

Die FITs[18] entsprechen im Wesentlichen einer PVSS-basierten Steuersoftware für jeweils eine Gerätefamilie. Damit bilden sie das PVSS-Pendant zu den bei kommerziellen Geräten häufig mitgelieferten Standalone-Tools, die sich nicht direkt in PVSS einbinden lassen. Ein FIT für ein bestimmtes Gerät unterstützt im Idealfall dieselbe Funktionalität wie das vom Hersteller der Hardware gelieferte eigenständige Programm.

Bei der Programmierung eines FITs muss der Entwickler demnach versuchen, alle Eigenheiten der Hardware in der Software zu berücksichtigen.

Besonders wichtig ist dabei der spätere Einsatzzweck der Software. Da die FITs die Grundlage der anderen Softwarekomponenten bilden (dazu mehr in 4.2.3 und 4.2.4), soll deren Konzept möglichst intuitiv und einheitlich sein.

Jeder FIT besteht im Wesentlichen immer aus den beiden Komponenten „Integration“ und „Control“, die über ein eigenes GUI[19] erreichbar sind. Unter Integration sind dabei alle Panel (PVSS-Jargon für Fenster) zu finden, mit denen neue Geräte dieser Familie in das System integriert werden können, während der Control-Teil die eigentlichen Panel für die Steuerung und Überwachung (also die Kontrolle) der Hardware zur Verfügung stellt.

Momentan werden folgende FITs genutzt:

- FIT_Iseg-HV (Iseg HV Crates)
- FIT_CAN-ELMB (verschiedene ELMB-basierte Typen, darunter: BBM, BBIM, SC-OLink, Regulator-Station,...
- FIT_Interlock (Logic-Units)
- FIT_Wiener-LV (Wiener LV Crates)

Es ist an dieser Stelle wichtig zu erwähnen, dass die FITs, genau wie die Standalone-Tools der Hersteller nicht „wissen“, welchen Teil des Detektors sie versorgen. Für einen FIT handelt es sich immer um einen bestimmten Power-Supply Kanal, der mit Hilfe seiner Position in einem Rack/Crate usw. identifiziert wird. Gleiches gilt für Temperatur- und andere Sensoren. Man spricht in diesem Zusammenhang von funktionaler Darstellung.

[18] **F**rontend **I**ntegration **T**ool
[19] **G**raphical **U**ser **I**nterface

4.2.3 Die System Integration Tools

Vorab zwei wichtige Begriffsdefinitionen, die in den folgenden Kapiteln benötigt werden:

Begriffsdefinition "funktional" / "geographisch"

funktional: Unter der funktionalen Gliederung von Komponenten versteht man die Gliederung gemäß des Gerätetyps bzw. der Zugehörigkeit zu bestimmten Gerätegruppen (beispielsweise „IsegHV“).

geographisch: Unter dem Begriff „geographisch“ versteht man die der Detektorgeographie folgende Gliederung von Komponenten. Dabei ist die Detektorgeographie durch die verwendeten Detektorteile definiert. Im Falle des ATLAS-Pixeldektors sind dies die vier Lagen „B-Layer“, „Layer1“, „Layer2“ und „Disk“, sowie die darunter liegenden Hierarchieebenen „Bi-Stave/Bi-Sector“, „Half-Stave/Sector“ und „Module“.

Obwohl der Detektor ausschließlich mit Hilfe der FITs betrieben werden könnte, wäre dies doch sehr aufwändig. Ohne das Wissen über die Verkabelung eines Power-Supply-Kanals oder die geographische Position eines Messwertes ist dieser Wert nur eingeschränkt von Nutzen.

Um einen einfachen und intuitiven Betrieb des Detektors für einen mit der Verkabelung nicht vertrauten Shifter zu ermöglichen, existiert daher der SIT[20]. Dieser ermöglicht die virtuelle Verkabelung des gesamten Detektors mit den verschiedenen DCS-Hardwarekomponenten durch den Einsatz spezieller XML[21]-Dateien.

Die Verbindung zwischen einer in Form eines Datenpunktelementes vorhandenen Hardwarekomponente, wie beispielsweise eines BBIM-Temperaturkanals mit der entsprechenden Detektoreinheit, wird dabei über "Aliase" realisiert. Abbildung 4.7 zeigt eine gemäß dem geographischen Detektorlayout (siehe Abschnitt 3.1) aufgebaute Baumansicht des Pixeldetektors. Die gelb eingezeichneten Datenpunkte stellen die Verbindung zu den Realdaten der Prozessparameter her, während jeder dieser Datenpunkte über den Alias-Zweitbezeichner eine Position innerhalb der geographischen Detektorhierarchie zugewiesen bekommt.

Im Anschluss an die erfolgte Verkabelung können die eingesetzten PVSS-Panels dem Beispiel der Abbildung folgend über den Alias "$L2\_B26\_S2\_A7\_TOpto$" auf den BBIM-Kanal "$BBIM05D.Ibox2.Temp8$" zurückgreifen, hinter dem sich die gewünschte Optoboard-Temperatur verbirgt.

[20] **S**ystem **I**ntegration **T**ool
[21] E**x**tensible **M**arkup **L**anguage

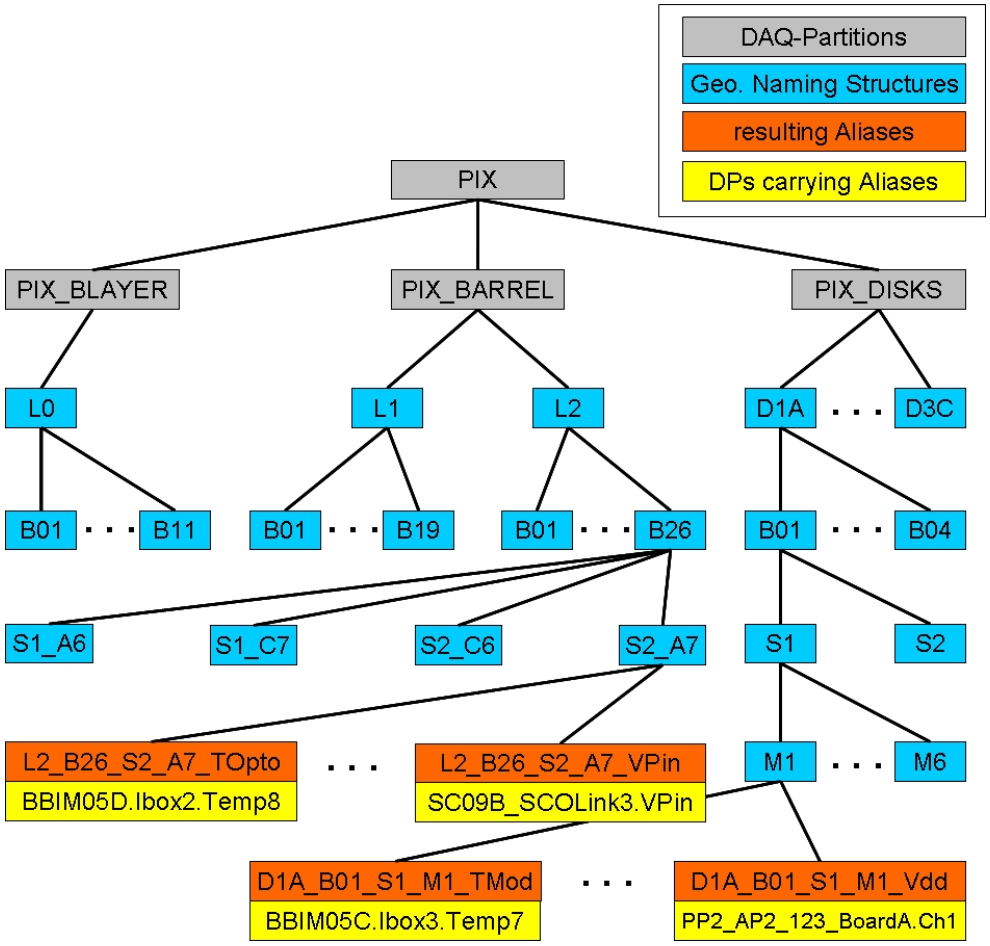


Abbildung 4.7: *Schematische Darstellung des Alias-Mechanismus.*

Die angezeigte Temperatur ist also beispielsweise nichts anderes, als der Wert des verbundenen BBIM-Temperaturkanals aus einem der FITs. Ebenso verhält es sich mit den Kontrollparametern, wie z.B. Setzspannungen. Diese können direkt über die DCS-Panel geändert werden, ohne dass der Nutzer wissen muss, welchen FIT-Kanal er dazu zu bedienen hätte.

Neben diesen offensichtlichen und notwendigen Erleichterung der Bedienbarkeit kümmert sich der SIT auch um das Archivieren der Daten, also deren Speicherung in speziellen Archiven. Dazu werden alle wichtigen Prozessparameter, also insbesondere die Werte der Module, Optoboards usw. in eine SQL[22]-Datenbank namens “Conditions-DB[23]"geschrieben, die zentral vom CERN verwaltet wird.

Neben der Conditions-Datenbank zur Archivierung der Prozessparameter der Pixeldetektors existiert eine weitere für den SIT relevante Datenbank, die Verkabelungs-Daten

[22] **S**tructured **Q**uery **L**anguage
[23] **D**ata **B**ase

des Pixeldetektors enthält und dementsprechend Connectivity-DB genannt wird. Diese enthält die vollständigen Verkabelungs-Informationen des Pixeldetektors und seiner Peripherie und wird daher vom SIT genutzt, um die Verbindung zwischen der funktionalen und geographischen Struktur herzustellen. Skripte seitens der DB-Software wandeln die Verkabelungsdaten in die bereits erwähnten XML-Konfigurationsdateien um, woraufhin diese vom SIT verarbeitet werden können.

4.2.4 Die Finite-State-Machine

Obwohl mit dem SIT bereits eine gewisse Möglichkeit besteht, den Pixeldetektor zu betreiben, darf nicht vergessen werden, dass auch dieser nicht dafür ausgelegt ist, 1744 Detektormodule in einem realistischen Schichtbetrieb zu verwalten.

Während des normalen Betriebes ist es ja gerade wünschenswert, die einzelnen Betriebsparameter nicht alle selbst zu setzen und zu überwachen, sondern den gesamten Detektor mit Hilfe weniger aussagekräftiger Befehle zu bedienen (SWITCH_ON, RECOVER, SWITCH_OFF, ...) und einen "raschen Überblick" zu erhalten.

Genau dies ermöglicht die FSM[24], die aus den tausenden Betriebsparametern des Detektors einige wenige Zustände ableiten kann.

Wichtigste Voraussetzung für die FSM ist dabei ein bereits funktionierender SIT, der die Grundlage für die FSM bildet, so wie die FITs die Grundlage für den SIT bilden.

Während alle bisher vorgestellten Softwarekomponenten nur für die DCS-Experten gedacht sind, ist die FSM die eigentliche Schnittstelle für den Shifter. Dieser hat jederzeit die Möglichkeit sich die Details in den von der FSM angezeigten Panels anzeigen zu lassen.

Neben dieser wichtigen Automatisierungsfunktion der FSM, stellt sie auch die Schnittstelle zur Datennahmesoftware des Pixeldetektors, DAQ[25], dar. Über den DDC[26]-Mechanismus (siehe Abschnitt 4.2.5) tauschen die FSMs auf DAQ- und DCS-Seite Informationen aus, wobei die DAQ-FSM als Master fungiert und Kommandos an die DCS-FSM weiterleitet. Darüberhinaus bildet die FSM auch die Schnittstelle zu der zentralen ALTAS-DCS-Kontrollstation, so dass der ATLAS-Detektor mit den anderen Subdetektoren synchronisiert werden kann.

Die von ATLAS-DCS definierte FSM-Architektur beschreibt den Detektor dabei durch FSM-Objekte, deren Zustände und durch Befehle, die den Zustand einzelner oder mehrerer dieser Objekte ändern können. Damit Teile eines Detektors einzeln betrieben und Fehler besser eingegrenzt werden können, besitzt die ATLAS-Pixel-DCS-FSM eine hierarchische Baumstruktur. Jeder der Knoten und jedes der Blätter dieses Baumes besitzt dabei einen

[24] **F**inite **S**tate **M**achine
[25] **D**ata **Acq**uisition
[26] **D**AQ **D**CS **C**ommunication

FSM-Zustand (siehe die folgende Begriffsdefinition). Dabei ist zwischen den Knoten, den "Control-Units", und den Blättern, den "Device-Units", des Baumes zu unterscheiden:

- **Device-Units:** *Diese stellen die Verbindung zu den in Form von Datenpunktelementen vorliegenden Realdaten, wie Temperaturen, Spannungen und Strömen her. Eine Device-Unit ermöglicht es darüber hinaus, die Informationen mehrerer Werte zu einem gemeinsamen Zustand zu bündeln. Device-Units sind die Blätter des Baumes und damit die unterste Hierarchieebene der FSM.*
- **Control-Units:** *Control-Units vermögen die Zustände mehrerer "Kinderknoten" zu einem gemeinsamen Zustand zu verbinden. Dabei ist es unerheblich, ob es sich bei den Kindern selbst um Knoten oder Blätter handelt. Daher finden sich Control-Units auf allen Hierarchieebenen der Pixel-FSM wieder, die keine direkte Verbindung zu Datenpunktelementen besitzen.*

Gemäß der ATLAS-DCS Definition wird der oben erwähnte Zustand einer Control- oder Device-Units durch den Einsatz zweier unterschiedlicher IST-Zustände abgebildet[27]. Unglücklicherweise ist die offizielle Namensgebung der beiden Zustände mit den Bezeichnungen "STATE" und "STATUS" etwas unintuitiv gewählt, so dass hier beide Begriffe genauer definiert werden sollen, da sie im Folgenden wichtig sind:

Begriffsdefinition "State" / "Status"

STATE: Der STATE entspricht dem Betriebsmodus einer Detektoreinheit. Grundsätzlich existieren auf allen Hierarchieebenen die Betriebsmodi/States "READY", "NOT_READY" und "UNKNOWN". Weitere States können je nach Einsatzgebiet hinzugefügt werden.

STATUS: Der STATUS einer Detektoreinheit spiegelt deren Befindlichkeit oder Alarmzustand wieder. Im Gegensatz zu den erweiterbaren States sind genau vier Alarmzustände/Status "OK", "WARNING", "ERROR", "FATAL" vorgegeben. Weitere Status[a] sind nicht erlaubt.

[a] Auch wenn es verwirrend erscheint, aber "Status" ist der grammatikalisch korrekte Plural (U-Deklination) des lateinischen "Status = Zustand, Befund, Bestand, Verfassung" und wird daher im weiteren auch so benutzt. Ob es sich um den Singular oder den Plural handelt, ist dem Kontext zu entnehmen.

Der Betrieb der FSM sieht dabei so aus, dass auf einer oberen Hierarchieebene Kommandos, wie etwa der Befehl "GOTO_READY", abgesetzt werden, die dann der Hierarchie der Knoten folgend nach unten weiterpropagiert werden. Befehle können von den

[27] Viele Beispiele aus anderen Bereichen vergleichen hingegen SOLL- und IST-Zustände.

einzelnen Hierarchiestufen immer nur an ihre Kinder weitergegeben werden. Im Gegensatz dazu können die Status nur von realen Betriebsparametern des Detektor generiert werden, weshalb diese sich von den Device-Units über ihre Eltern-Control-Units nach oben ausbreiten.

Erhält eine Device-Unit den Befehl, seinen State zu ändern, so wird dies intern durch PVSS-Steuerskripte realisiert. Alle im Rahmen eines solchen State-Übergangs relevanten Prozessparameter werden dabei aus eine Konfigurationsdatenbank bezogen, was beispielsweise die Feinabstimmung einzelner Setzspannungen pro Detektormodul ermöglicht. Diese "Configuration"-DB enthält alle Setzspannungen und Ströme, sowie alle zur Beurteilung des Status notwendigen Mess-Grenzwerte des Pixeldetektors.

Der Aufbau der FSM sieht dabei vor, dass sich Zweige eines Baumes nahezu beliebig aus dem Betrieb der restlichen FSM ausklinken lassen. Über verschiedene Mechanismen ist es damit möglich, das Weiterleiten von Kommandos bzw. Alarmleveln zu unterbinden. Weiterhin lassen sich damit einzelne Zweige des Hierarchiebaumes getrennt voneinander autark betreiben. Diesem Konzept folgend besteht die FSM des Pixeldetektors aus mehreren FSM-Partitionsrechnern, die jeweils einen Hauptzweig der Detektorhierarchie verwalten. Diesen übergeordnet kommt pro Subdetektor eine SCS[28] zum Einsatz, die die einzelnen FSM-Partitionsrechner miteinander verbindet und als direkter Ansprechpartner der globalen ATLAS-GCS[29] dient (siehe Abbildung 4.8).

Im Gegensatz zu den FSM-Partitionen, die Kommandos von den jeweiligen SCS-Rechnern erhalten, geben die SCS-Rechner selbst jeweils nur ihre State/Status-Werte an die globale ATLAS-GCS weiter, ohne von dieser Befehle entgegenzunehmen. Dieser Bruch ist notwendig, da die einzelnen Subdetektoren autark operieren können sollen und daher nicht von einer globalen DCS-FSM, sondern von ihrer zugehörigen Subdetektor-DAQ-FSM gesteuert werden.

4.2.5 DAQ-DCS-Kommunikation

Die im letzten Abschnitt erwähnten FSMs auf DAQ- und DCS-Seite sind miteinander synchronisiert, um zu gewährleisten, dass der von der DCS-FSM betriebene Detektor bereit für die von der DAQ-FSM gesteuerte Datennahme ist. Die DAQ-FSM fungiert dabei als Master und setzt somit Befehle an die DCS-FSM ab, die wiederum ihre Zustandswerte (State und Status) mitteilt.

Beide Systeme werden über einen als DDC bezeichneten Mechanismus miteinander verbunden. Neben der wichtigen Hauptaufgabe der Synchronisation der beiden FSMs hat DDC noch eine Reihe weiterer Aufgaben im Rahmen der Kommunikation zwischen DAQ und DCS zu erfüllen. So gibt es einige DCS-Parameter, wie beispielsweise die BOC-

[28] **S**ubdetector **C**ontrol **S**tation
[29] **G**lobal **C**ontrol **S**tation

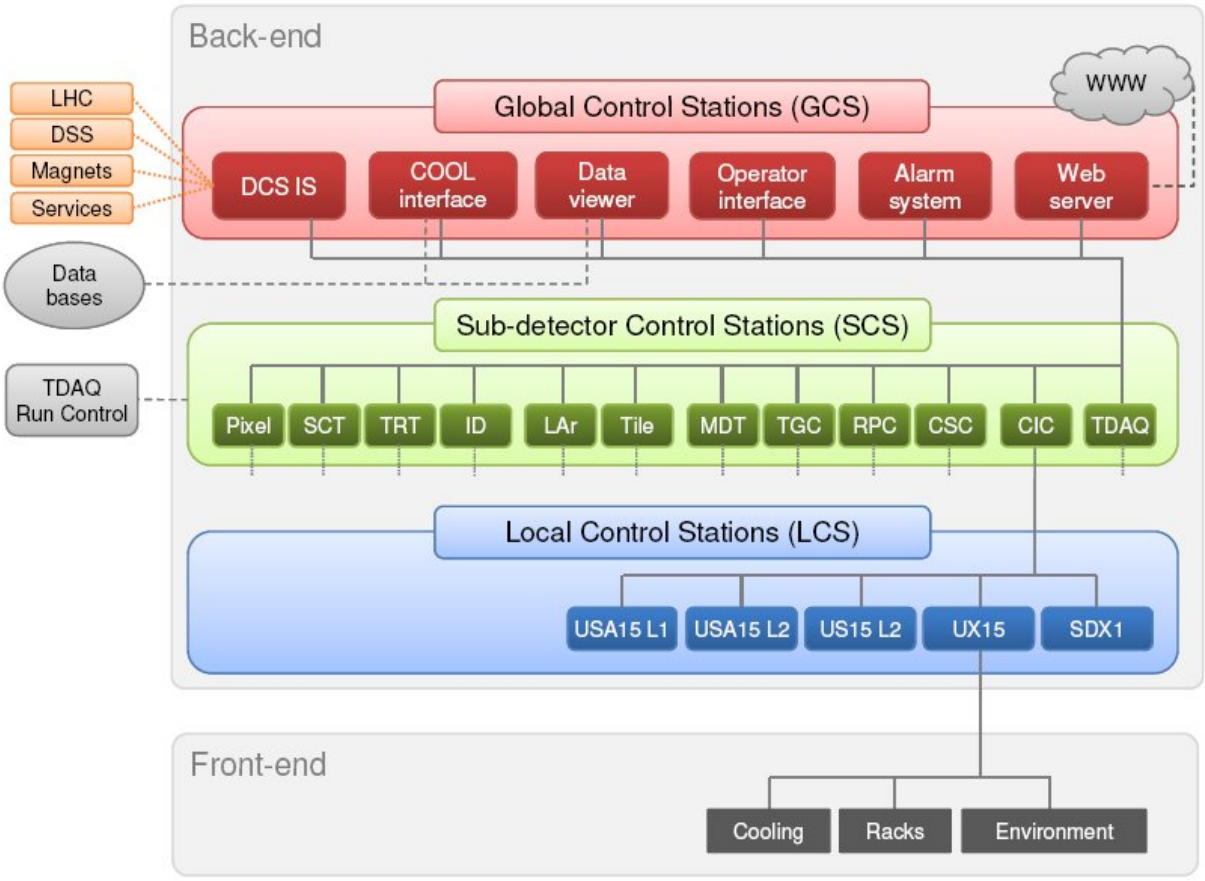


Abbildung 4.8: *schematische Darstellung der ATLAS-FSM-Hierarchie[3]*

Temperaturen, die nur von DAQ-Sensoren ausgelesen werden können, weshalb diese dann über den DDC-Mechanismus an das Detektorkontrollsystem weitergereicht werden müssen. Umgekehrt gibt es Fälle, wie beispielsweise die Kalibration eines Optoboards, in denen der eigentliche Scan der auf der DAQ-Seite durchgeführt wird, von der Einstellung entsprechender DCS-Werte abhängt.

Da die DCS-Rechner zum Teil auf Grund der Microsoft-proprietären OPC-Architektur auf den Einsatz von Windows-Betriebssystemen angewiesen sind, die DAQ-Rechner allerdings durchweg mit Linux betrieben werden, kommt ein flexibles Kommunikationsprotokoll für den Austausch der Daten und Kommandos zum Einsatz. Dieses soll im folgenden Abschnitt in Folge seiner Wichtigkeit im Zusammenhang mit dieser Arbeit näher beschrieben werden.

4.2.5.1 Distributed Information Management

Das DIM[30]-Protokoll ist der DDC zugrunde liegende Kommunikationsmechanismus. Abbildung 4.9 zeigt die wesentlichen Komponenten dieser Software. Als Client/Server-Architektur verfügt DIM über Server, die Daten zur Verfügung stellen und Clients, die diese abrufen.

Da DIM, wie der Name schon impliziert, auf den Einsatz in Systemen mit mehreren Rechnern ausgerichtet ist, registriert jeder Server (denn es kann beliebig viele Server geben) die durch ihn bereitgestellten Daten in einem "Name Server". Dort werden die Daten unter dem Synonym "Services" geführt, das auch in der Abbildung 4.9 verwendet wird. Dass es sich bei den Services nicht nur um vom Client lesbare Daten, "Information Services" oder kurz "Infos" handeln muss, sei zunächst vernachlässigt.

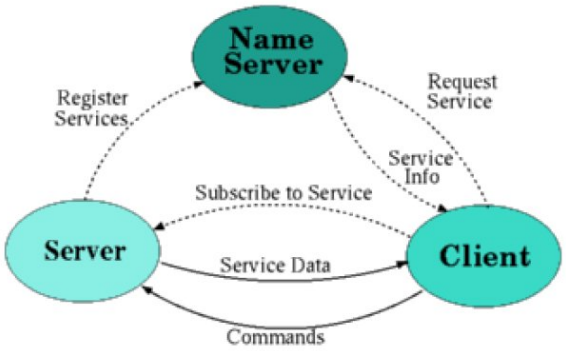


Abbildung 4.9: *Grundprinzip des DIM-Mechanismus [22]*

Sobald ein Client auf bestimmte Daten zugreifen möchte, sendet er einen "Service Request" an den Name Server. Dieser teilt dem Client dann mit, auf welchem der vom Name Server verwalteten Rechner der Client die gewünschten Daten finden kann. Nach dieser initialen Kommunikation treten Client und Server direkt miteinander in Kontakt und der Name Server wird nicht mehr benötigt. Der Client hat an dieser Stelle die Möglichkeit, Daten entweder lediglich einmalig abzurufen und die Kommunikation mit dem Server danach wieder zu beenden, oder sich vom Server bei Wertänderung informieren zu lassen. Das einmalige Holen der Daten wird in DIM allgemein als "GET"-, das Abonnieren als "SUBSCRIBE"-Mechanismus bezeichnet.

Neben der Bereitstellung von durch den Client abrufbaren Daten kann ein DIM-Server auch Kommandos zur Verfügung stellen, über die die Clients bestimmte Aktionen im Server auslösen können. Ein Beispiel für ein solches Kommando ist das Setzen einer Spannung durch einen Client. Da Befehle auch fehlschlagen können, ist ein Rückgabewert als Indikator für das Gelingen einer Befehlsausführung in den meisten Fällen notwendig. Während Information Services vom Client nur gelesen und ihre Gegenstücke, die "Command Services" von diesen nur geschrieben werden können, ermöglicht die Bündelung der beiden Typen in RPC[31]-Services eine bidirektionale Kommunikation zwischen Client und Server. Die Gesamtheit aus Info-, Command- und RPC-Objekten nennt man auch DIM-Items.

[30] **D**istributed **I**nformation **M**anagement

[31] **R**emote **P**rocedure **C**all. RPCs gibt es in vielen Programmiersprachen, wobei im Rahmen dieser Arbeit jedoch ausschließlich der DIM-eigene RPC-Mechanismus unter dem Begriff "RPC" gemeint ist.

Ein RPC besteht dabei immer aus drei Service-Items:

- **RPC-Parameter**: *Viele Kommandos benötigen weitere Parameter zur Spezifikation der korrekten Aktion. Diese können vom Client in den Command-Service RPC-Parameter geschrieben werden*
- **RPC-In** oder **Trigger**: *Dieses Command-Service-Bit wird von den Clients genutzt, um die Ausführung des Befehls auf Seiten des Servers zu starten. Da die interne Ausführung des Befehls auf Serverseite oftmals schneller erfolgt als eine einzelne DIM-Transaktion, ist der Parameter vor dem Auslösen des Triggers zu setzen.*
- **RPC-Out** oder **Response**: *Nachdem der Server die notwendigen, durch den Trigger ausgelösten Schritte unter Berücksichtigung der gegebenen Parameter ausgeführt hat, setzt dieser das Response-Bit, um dem damit verbundenen (per "SUBSCRIBE") Client die Beendigung des Kommandos zu signalisieren.*

In den meisten Fällen werden zusätzliche Info-Services verwendet, um Aussagen über die Qualität der Kommandoausführung zu liefern (Errorcodes, Rückgabewerte).

Der DDC-Mechanismus des Pixeldetektors ist so implementiert, dass sowohl DCS als auch DAQ gleichzeitig als DIM-Server und Client agieren. Auf DCS-Seite kommt dabei ein PVSS-DIM-Manager zum Einsatz, der die notwendigen DIM-Funktionen direkt aus PVSS-Skripten heraus aufrufbar macht.

Da zwischen DAQ und DCS eine Vielzahl an möglichen Kommandos ausgetauscht werden, wurde der RPC-Mechanismus für den Pixeldetektor erweitert. Anstelle vieler einzelner RPC-Services, die jeweils immer nur für genau ein Kommando zur Verfügung stehen, nutzt das Pixeldetektor-DDC einen wesentlich flexibleren Mechanismus zur Nutzung der RPCs.

Dabei werden spezielle RPC-Datenpunkte (siehe Abbildung 4.10) von einem speziellen PVSS-Skript überwacht, das im Falle des Setzens des RPC-Triggers aufgerufen wird. Anstatt jedem RPC nur ein einziges Kommando zuzuordnen, wird das gesamte Kommando, inklusive Parametern in das Parameter-Item des RPCs geschrieben (etwa "SET_HIGHVOLTAGE D1A_B02_S1_M4 600V"). Dieser Parameter wird dann von dem erwähnten PVSS-Skript ausgewertet und in seine verschiedenen Bestandteile zerlegt, die dann entscheiden, welche Aktion ausgeführt werden muss. Da je nach gewünschtem Kommando verschiedene Rückgabewerte von

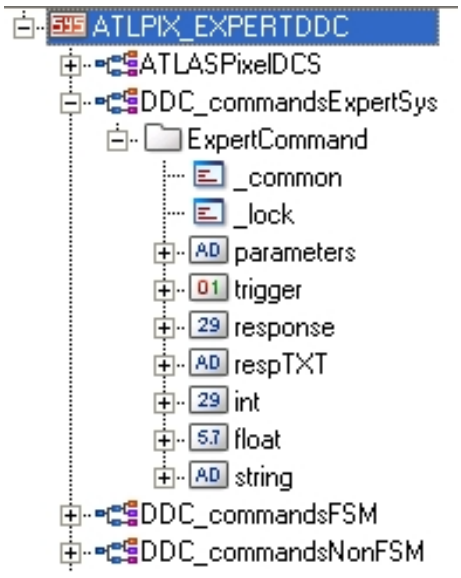


Abbildung 4.10: *Kommandodatenpunkt des Expertensystems.*

Interesse sind, gibt es zu jedem DDC-RPC mehrere Rückgabewert-Info-Services für Gleitkommazahlen, Zeichenketten usw.. Nachdem das PVSS-seitige Skript den Befehl abgearbeitet hat, wird das Resultat in den entsprechenden Rückgabe-Service geschrieben und das Response-Bit wird gesetzt, um den Client über die beendete Ausführung zu informieren.

Ausführliche Untersuchungen[41] auf vergleichsweise moderner Hardware haben gezeigt, dass bei einer Updaterate von 5 s bis zu 25000 Datenpunktelemente durch den PVSS-DIM-Server bereitgestellt werden können, bevor das System instabil wird. Berücksichtigt man, dass DIM ausschließlich für den Transport von Nachrichten und Kommandos konzipiert wurde, sowie die Tatsache, dass während des Betriebs des Pixeldetektors lediglich die Status-/State-Kombinationen der FSM-Device- und Control-Units übertragen werden sollten, scheint diese Zahl mehr als ausreichend[32]. Im Falle eines detektorweiten Interlocksignals, wie es durch I_DSS ausgelöst werden kann, würden sich jedoch alle Status-/State-Paare innerhalb kürzester Zeit ändern, was das System kurzzeitig an seine Belastungsgrenze bringen würde.

Eine zusätzliche, gravierende Belastung des DIM-Mechanismus ist daher von allen anderen Softwarekomponenten zu vermeiden. Insbesondere die Anzahl zusätzlicher DIM-Items ist auf ein Minimum zu reduzieren (siehe Abschnitt 8.2.1).

[32]Die FSM nutzt weniger als 3000 Device-/Control-Units mit jeweils zwei Werten, die sich im Normalfall selten ändern. Selbst wenn der gesamte Detektor an oder ausgeschaltet wird, ist die tatsächliche Datenrate weit geringer als die dadurch theoretisch möglichen 6000 Werte pro Sekunde, da die FSM selbst Zeit benötigt, um die notwendigen Übergänge zu vollziehen.

Kapitel 5

Expertensysteme

Bevor in Kapitel 8 auf das im Rahmen dieser Arbeit konzipierte Expertensystem für das Detektorkontrollsystem des Pixeldetektors eingegangen wird, folgt hier zunächst eine kurze Einführung in das Gebiet der Expertensysteme.

5.1 Definition eines Expertensystems

Expertensysteme bilden ein Untergebiet der Künstlichen Intelligenz. Während bereits in den 50er Jahren versucht wurde, Computerprogramme mit künstlicher Intelligenz, getreu dem Motto:

"Every aspect of learning or any other feature of intelligence can be so precisely described that a machine can be made to simulate it." [John McCarthy][1]

zu entwickeln, konnten diese ersten Versuche die an sie gestellten Erwartungen nicht erfüllen.

Da schnell erkannt wurde, dass das ambitionierte Ziel eines "General Problem Solvers" weder mit der damaligen Software, noch mit der zur Verfügung stehenden Hardware realisiert werden konnte, beschränkte man sich auf stark eingeschränkte Problemfelder. Ziel war es nunmehr, nicht mehr alle von einem intelligenten Wesen lösbaren Probleme mit Hilfe eines Computers zu lösen (General Problem Solver), sondern Programme zu entwickeln, die in einem eingeschränkten Wissensbereich Probleme identifizieren und lösen

[1]Das angeführte Zitat ist Teil des inzwischen berühmten "Dartmouth Vorschlages", der im Rahmen der für das Forschungsgebiet der Künstlichen Intelligenz wegweisenden Konferenz "Dartmouth Summer Research Conference on Artificial Intelligence" von John McCarthy und Kollegen unterbreitet wurde. Die im Sommer 1956 abgehaltene Konferenz gilt gemeinhin als Beginn der Künstlichen-Intelligenz-Forschung unter diesem Namen. John McCarthy entwickelte im Jahr 1960 die Programmiersprache LISP (**Lis**t **P**rocessor), die noch immer die Grundlage der in dieser Arbeit genutzten Logiksprachen bildet[42][1].

können. Solche Systeme werden Expertensysteme genannt und in einigen Gebieten bereits seit Jahrzehnten erfolgreich eingesetzt[2].

Die sich dabei stellende Aufgabe, aus gegebenen Daten auf die Ursache eines Problems zu schließen, kann mit verschiedenen Ansätzen gelöst werden. Die bis heute beliebteste und verbreitetste Form bilden dabei "regelbasierte Systeme", die die vorhandenen Daten mit Hilfe von "Produktionsregeln" auswerten und daraus Schlüsse ziehen. Solche Systeme werden daher auch als "Produktionssysteme" bezeichnet.

Ein weiterer Ansatz für die Realisierung eines Expertensystems besteht in der Verwendung künstlicher neuronaler Netze. Während die Vorstellung eines "denkenden", "intelligenten" Computers im Bereich der Expertensysteme sicherlich sehr reizvoll ist, fehlt in diesem Bereich oftmals die notwendige Voraussetzung für den Einsatz dieser Netze: die Möglichkeit das System lernen oder trainieren zu lassen. Dazu müsste ein neuronales Netz entweder über eine exakte Simulation des zu beschreibenden Expertengebiets, auch Wissensdomäne genannt, verfügen oder die Möglichkeit besitzen, anhand der Reaktionen des realen Systems zu lernen. Beides ist in vielen Fällen schwierig bzw. gefährlich zu realisieren.

Das in dieser Arbeit beschriebene DCS-Expertensystem stellt ein regelbasiertes System dar, weshalb sich die folgenden Abschnitte ausschließlich auf diesen Typ des Expertensystems konzentrieren. Da die meisten Expertensysteme auf Produktionssystemen beruhen und damit ebenfalls regelbasiert sind, werden diese Terme im weiteren Verlauf synonym genutzt[3]. "Expertensystem" bedeutet im folgenden also immer "regelbasiertes Expertensystem".

5.2 Aufbau eines Expertensystems

Jedes regelbasierte Expertensystem besteht aus den vier kooperierenden Hauptkomponenten "Regelbasis", "Datennahmekomponente", "Inferenzmaschine" und "Benutzerschnittstelle" (siehe auch Abbildung 5.1), die in den folgenden Abschnitten erklärt werden sollen.

5.2.1 Die Regelbasis

Die Regelbasis eines Expertensystems ist Teil der übergeordneten Wissensbasis, die von den vier Komponenten gemeinsam genutzt wird und die das gesamte problemspezifische Wissen speichert. Dies beinhaltet neben den aktuell vorliegenden Daten insbesondere die

[2]Das berühmteste Beispiel ist wohl das medizinische Diagnosesystem „Mycin", Stanford University, 1972.

[3]Insbesondere da der Term "Produktionssystem" im Bereich der Softwareentwicklung noch eine zweite Bedeutung hat, die ausdrückt, dass es sich bei einem bestimmten Rechner nicht um einen Entwicklungsrechner handelt (Gegenüberstellung von Produktion und Entwicklung).

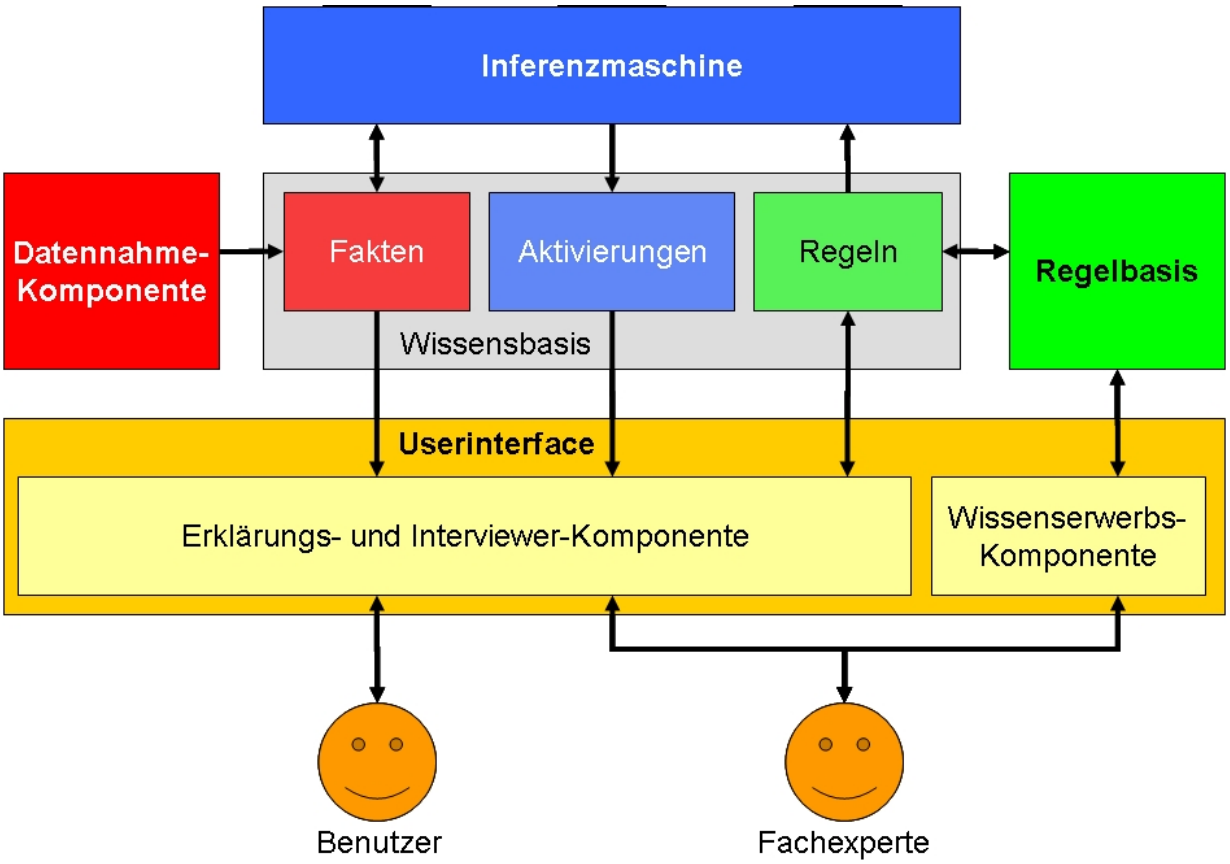


Abbildung 5.1: *Prinzipieller Aufbau eines Expertensystems*

notwendigen Regeln, um aus den in der Datennahmekomponente gesammelten Fakten mit Hilfe des Inferenzmechanismus (siehe 5.2.3) Schlussfolgerungen zu ziehen. Diese werden in der Regelbasis gespeichert, die rein anwendungsspezifisches Wissen enthält und daher für jedes Einsatzgebiet neu geschrieben werden muss.

Die Regeln bilden dabei das Wissen eines oder mehrerer menschlicher Fachexperten auf dem entsprechenden Gebiet ab und bilden damit die Wissensgrundlage des Expertensystems.

Abhängig von der genutzten Inferenzmaschine unterscheidet sich die Regelsyntax des Expertensystems. Allen Systemen gemein ist jedoch der prinzipielle Aufbau einer Regel in "Regelname", "Prämisse" und "Konklusion", wie in Abbildung 5.2 gezeigt.

Abbildung 5.2: *Allgemeiner Aufbau einer Regel*

Je nach Einsatzzweck und Anforderung, kann die Regelbasis dabei in Form von Textdateien oder Datenbanken vorliegen, was jedoch für die weitere Betrachtung unerheblich ist.

5.2.2 Die Datennahmekomponente

Die Datennahmekomponente bildet die direkte Verbindung zu den Realdaten des jeweiligen Problems. Die sich hier stellende Problemstellung ist oft rein technischer Natur, da eine Interfacemöglichkeit zwischen der Datennahmesoftware und dem Expertensystem gefunden werden muss. Dennoch hat die Datennahmekomponente einen wichtigen Einfluss auf den genutzten Inferenzmechanismus, da sie in vielen Fällen bestimmt, wieviele Daten mit welcher Frequenz berücksichtigt werden können.

Die von der Datennahmekomponente in das Expertensystem eingegebenen Daten werden gemeinhin als Fakten bezeichnet, bei denen es sich um einzelne Werte oder aber auch um komplexe Objekte handeln kann (siehe Abschnitt 5.2.3.1).

5.2.3 Die Inferenzmaschine

Den eigentlichen Kern des Expertensystems bildet die Schlussfolgerungs- oder auch "Inferenzkomponente"[4]. Diese verarbeitet die von der Datennahmekomponente bereitgestellten Daten mit Hilfe der in der Wissensbasis spezifizierten Regeln.

Dabei unterscheidet man prinzipiell das Abgleichen der vorhanden Daten mit der Gesamtheit der vorhandenen Regeln (Abschnitt 5.2.3.1), und die darauffolgende Priorisierung der passenden Regeln (Abschnitt 5.2.3.2).

Je nach Problemstellung kommen dabei verschiedene Inferenzstrategien für den Datenfluss und die Regelauswertung zum Einsatz:

- Datenfluss:

  Vorwärtsverkettung: *Die Schlussfolgerungen werden durch Auswertung aller genommenen Daten gezogen. Daher werden vorwärtsverkettete Inferenzmechanismen auch „datengetrieben" genannt.*

  Rückwärtsverkettung: *Hier werden die Daten selektiv, anhand der offensichtlichen Symptome unter Annahme einer Hypothese genommen. Diese Daten werden dann mit der Hypothese verglichen.*

[4]Der lateinische Begriff "Inferenz" (von *infero*: folgern, schließen) wird in diesem Zusammenhang für die logikbasierte Schlussfolgerung eines Computerprogramms verwendet.

- Regelauswertung:

  Probabilistisches Schließen: *Hierbei werden die einzelnen Schlussfolgerungen mit mathematischen Wahrscheinlichkeiten gewichtet.*

  Unsicheres Schließen: *Im Gegensatz zum probabilistischen Schließen werden die Gewichte nicht streng als mathematische Wahrscheinlichkeiten, sondern nur als Wichtungsfaktoren gewertet. Man geht dabei oft zu Wahrscheinlichkeitsbereichen oder eher qualitativen Aussagen wie „gesichert“, „vielleicht“ oder „unwahrscheinlich“ über.*

In vielen Fällen kommen auch Kombinationen der oben genannten Inferenzstrategien zum Einsatz.

5.2.3.1 Der Rete-Algorithmus

Unabhängig von der gewählten Inferenzstrategie ist die zentrale Aufgabe der Inferenzmaschine der Abgleich der durch die Datennahmekomponente in das System eingefügten Fakten mit den in der Regelbasis vorhandenen Regeln. Da die Wissensbasis aus einer großen Zahl komplexer Regeln und Fakten bestehen kann, ist es notwendig einen möglichst effizienten Algorithmus zu nutzen, um zu bestimmen, welche der vorhandenen Regeln zu den Fakten passen.

Ein naiver Lösungsansatz besteht darin, die einzelnen Regeln der Reihe nach jeweils mit allen vorhandenen Fakten abzugleichen und diesen Prozess zu wiederholen, sobald ein neues Faktum in die Wissensbasis eingefügt, bzw. der Wert eines bestehenden Faktums geändert wurde. Obwohl dieser Algorithmus prinzipiell funktioniert, ist er in einem System mit vielen Regeln und vielen, sich häufig ändernden Fakten sehr rechenaufwändig und wird daher vermieden.

Ein wesentlich leistungsfähigerer Ansatz, der sich daher auch in den meisten modernen Produktionssystemen durchgesetzt hat, ist der “Rete”-Algorithmus. Im Gegensatz zum angedeuteten naiven Algorithmus nutzt dieser mehrere Optimierungen, um den Inferenzvorgang signifikant zu beschleunigen.

Der Name Rete[5] bezieht sich dabei auf die netzartige Inferenzstruktur, die von den Fakten während des Inferenzprozesses durchlaufen wird und für die sich ein Beispiel in Abbildung 5.3 findet. Dieses Netz besteht aus fünf unterschiedlichen Knotenarten, die ein eingehendes Faktum dabei nur dann an die nachfolgenden Knoten weiterleiten, wenn die durch sie überprüfte Eigenschaft durch das Faktum erfüllt wird.

Das Rete-Netz besitzt dabei genau einen Eingangsknoten, der gemeinhin auch als “Rete-Node” bezeichnet wird. Dieser dient als Schnittstelle zur Datennahmekomponente des Expertensystems und stellt keine weiteren Bedingungen an die einzelnen Fakten.

[5] lat.: Netz

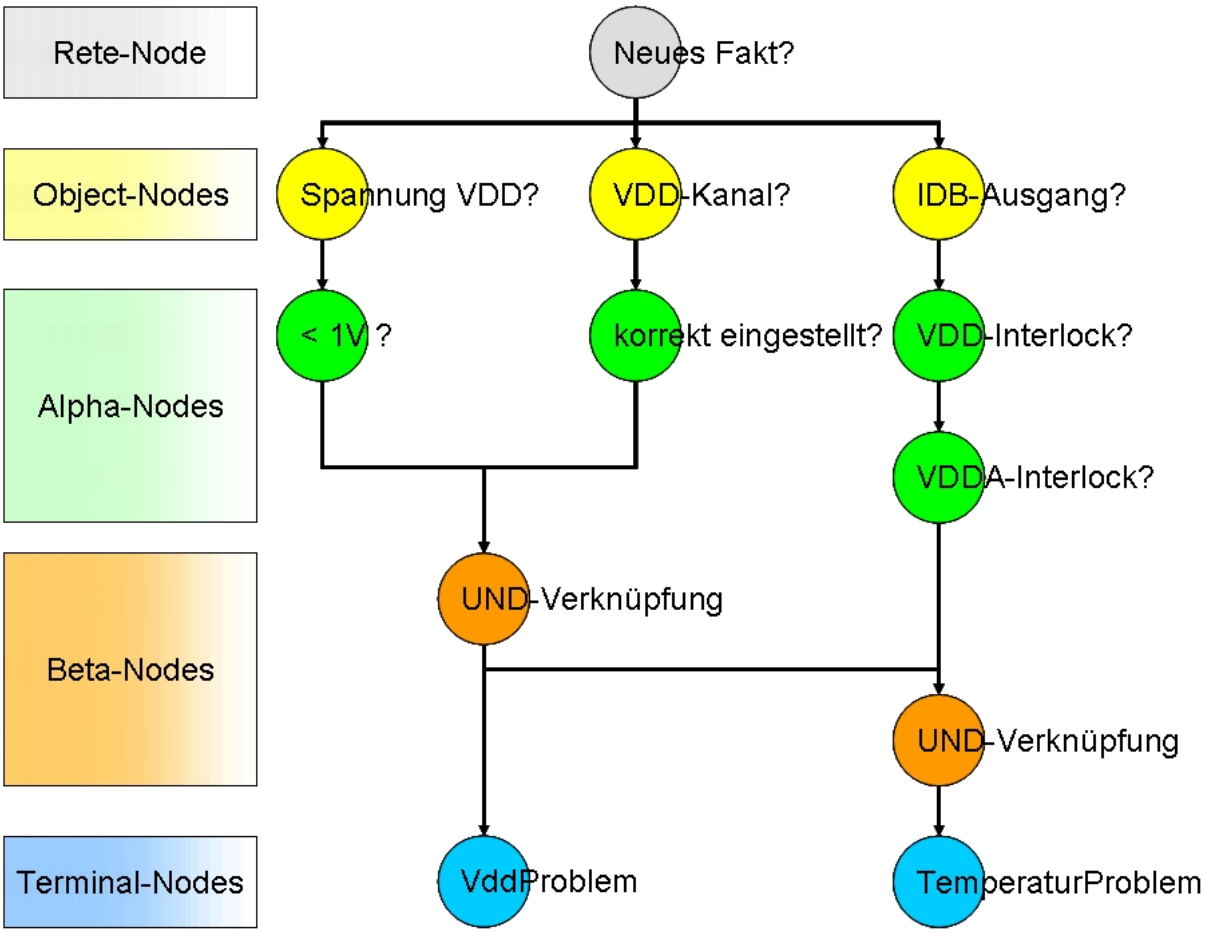


Abbildung 5.3: *Beispiel des Rete-Netzes für die beiden Regeln 5.4(a) und 5.4(b).*

An dieser Stelle ist anzumerken, dass die erwähnten Fakten, je nach Inferenzmaschine unterschiedlich repräsentiert werden. Die einfachste und auch oft genutzte Repräsentationsmöglichkeit besteht in einem einfachen Tupel aus Faktenname und Faktenwert. Objektorierte Inferenzmaschinen hingegen bieten meist die Möglichkeit, ganze Objekte, also Instanzen einer Klasse, als einzelne Fakten aus dem aufrufenden Programmcode zu übernehmen und auf deren Parameter direkt zuzugreifen.

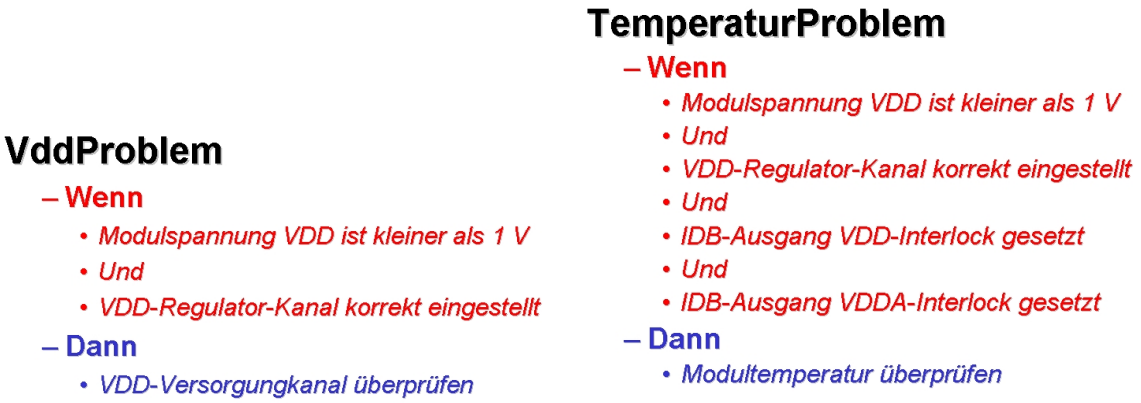
Nachdem neue Fakten den Rete-Knoten durchlaufen haben, kommen in objektorientierten Implementationen des Algorithmus “Object Nodes” zum Einsatz, die den Objekt-Typ des Faktums überprüfen und analog zu einer Weiche eine erste Festlegung der Fakten auf bestimmte Teilbereiche des Rete-Netzes vornehmen. In den darauf folgenden “Alpha-Nodes” werden einzelne Eigenschaften der durch die Object-Nodes vorselektierten Fakten-Objekte untersucht, bevor die “Beta-Nodes” solche Kandidaten-

Fakten miteinander logisch[6] verknüpfen, die alle davor liegenden Alpha-Knoten passiert haben. Eventuell erreichen ein oder mehrere Fakten schließlich die "Terminal-Nodes", die jeweils mit einer Regel assoziiert sind und erzeugen damit deren "Aktivierungen".

Begriffsdefinition "Aktivierung"

Unter einer **Aktivierung** versteht man das Markieren einer zum gegebenen Datensatz passenden Regel durch die Inferenzmaschine.

Abbildung 5.4 zeigt zwei einfache Beispielregeln, deren dazugehöriges Rete-Netz in Abbildung 5.3 dargestellt ist. Nachdem die Fakten den initialen Rete-Knoten durchlaufen haben, werden sie in die drei beteiligten Objekttypen eingeteilt. Im Falle der "VddProblem"-Regel wird demnach untersucht, ob es Fakten vom Typ "Modulspannung VDD" oder "VDD-Kanal" gibt. Sollte dies der Fall sein, wird durch die erwähnten Alpha-Knoten überprüft, ob diese Fakten bestimmte Eigenschaften erfüllen, beispielsweise, ob die Spannung unterhalb von 1 V liegt. Sollte dies der Fall sein, wird das entsprechende Faktum an nachfolgende Alpha- oder Beta-Knoten weitergegeben. Um eine überflüssige, doppelte Überprüfung eines bereits eingegebenen Faktums zu vermeiden, wird jedes eingehende Faktum zudem im "Alpha-Memory" des Alpha-Knotens zwischengespeichert.



(a) Vereinfachtes Beispiel einer VDD-Regel.

(b) Vereinfachtes Beispiel einer Temperatur-Regel.

Abbildung 5.4: *Zwei einfache Beispielregeln mit mehreren Prämissen und jeweils einer Konklusion.*

Im Gegensatz zu Alpha-Knoten besitzen Beta-Knoten generell zwei Eingänge und leiten nur dann die anliegenden Fakten weiter, wenn die Kombination aus linkem und rechtem Eingang die geforderte logische Verknüpfung (im Beispiel ein UND) erfüllt. Auch hier kommen Eingangszwischenspeicher zum Einsatz, um zu verhindern, dass bereits vorhandene, identische Fakten erneut überprüft werden. Die Faktenkombinationen, die der

[6] Dabei können, je nach Inferenzmaschine, alle gängigen logischen Verknüpfungen wie "UND", "ODER", "NICHT", "KEIN" etc. implementiert sein.

Überprüfung durch den letzten Beta-Knoten standhalten können, werden dann an den entsprechenden Terminal-Knoten weitergeleitet und erzeugen somit die Aktivierung der entsprechenden "VddProblem"-Regel. Auch die Terminal-Nodes besitzen einen Zwischenspeicher, der in der Lage ist, die Faktenkombinationen die zu den einzelnen Aktivierungen geführt haben, zu speichern.

Während die erwähnte "VddProblem"-Beispielregel bereits einige wesentliche Ideen des Rete-Algorithmus verdeutlicht hat, so wird ein weiterer wichtiger Optimierungsansatz sichtbar, wenn man die "TemperaturProblem"-Regel betrachtet. Wie schon aus dem vereinfachten Regelcode sichtbar, teilt diese mehrere Prämissen mit der "VddProblem"-Regel. Dies wird im Rete-Algorithmus ausgenutzt, indem die entsprechenden Alpha- und Beta-Nodes von beiden Regeln geteilt und somit pro Fakt nur einmal durchlaufen werden müssen. Im Beispiel kann man sofort erkennen, welchen Vorteil dieses "Node-Sharing" in Zusammenarbeit mit der Zwischenspeicherung der Fakten an den einzelnen Knoten hat. In einer Situation, in der bereits Fakten existieren, die das Vorhandensein eines Vdd-Problems untermauern, könnte das zusätzliche Auftauchen eines entsprechenden Interlocksignals auf ein Temperaturproblem hindeuten. In einem solchen Fall muss die Inferenzmaschine jedoch lediglich die beiden Interlockbedingungen überprüfen, da der "linke" Zwischenspeicher des letzten Beta-Knotens bereits die "Partial-Matches", also die Teiltreffer der vorhergehenden, gemeinsam genutzten Knoten enthält.

Zusammenfassend bietet der Rete-Algorithmus folgende wichtige und automatisch vorgenommene Verbesserungen gegenüber einem naiven Pattern-Matching-Verfahren:

- Zwischenspeicherung: *Jeder Knoten merkt sich die Fakten, die an ihm anliegen und damit die Fakten, die die vorhergehenden Knoten erfolgreich passiert haben.*
- Einmalige Überprüfung: *Jedes Faktum wird pro Knoten maximal einmal überprüft. Erst eine Wertänderung des Faktums bewirkt eine erneute Überprüfung.*
- Node-Sharing: *Der Rete-Algorithmus untersucht ALLE in der Regelbasis vorhandenen Regeln auf gemeinsam genutzte Knotenstrukturen. Sollten solche Strukturen vorhanden sein, werden entsprechende Bereiche des Rete-Netzes gemeinsam genutzt und nicht für jede Regel getrennt ausgewertet.*

Während der vorgestellte Rete-Algorithmus den skizzierten, naiven Ansatz in der Geschwindigkeit je nach Anwendung mehr oder weniger deutlich schlägt, sei angemerkt, dass dieses Plus an Performance durch einen höheren Speicherbedarf erkauft wird[7].

Alle im weiteren betrachteten Inferenzmaschinen nutzen daher den Rete-Algorithmus.

[7]Bei der seit Jahren üblichen und stetig wachsenden Speicherausstattung der eingesetzten Rechner ist allerdings gewährleistet, dass die enorme Anzahl an Fakten, die notwendig wäre, den Speicher zu füllen, im Falle des naiven Ansatzes zu einem sehr langsamen oder gar nicht mehr reagierenden System führen würde.

5.2.3.2 Konflikte

Da im Allgemeinen mehr als eine Schlussfolgerung aus den Daten gezogen werden kann, oder mit anderen Worten mehr als eine Regel aktiviert wurde, muss mit Hilfe einer Konfliktlösungsstrategie eine Ordnung innerhalb der "Agenda" erzeugt werden, damit die entsprechenden Regeln "gefeuert" und die damit verbunden Konklusionen ausgeführt werden können. Dazu dienen beispielsweise Filtermechanismen, die Regeln auf Grund bestimmter Faktoren, wie beispielsweise "Aktualität der Daten" oder "Spezifität", gewichten.

Begriffsdefinition "Agenda" und "feuern einer Regel"

Die Menge aller durch die Fakten der Wissensbasis erzeugten Aktivierungen nennt man **Agenda**.
Die Auswahl und Ausführung einer auf der Agenda stehenden Regel wird als **feuern** der Aktivierung bezeichnet.

Im Jargon der Expertensysteme wird dabei die Regel zuerst ausgeführt oder gefeuert, die nach der Konfliktlösung die höchste Priorität besitzt (siehe Abbildung 5.5). Allerdings ist es abhängig von der Implementation des konkreten Expertensystems, ob nur die Aktivierung mit der höchsten Priorität gefeuert wird, oder ob gegebenenfalls weitere Regeln mit niedrigerer Priorität ausgeführt werden können.

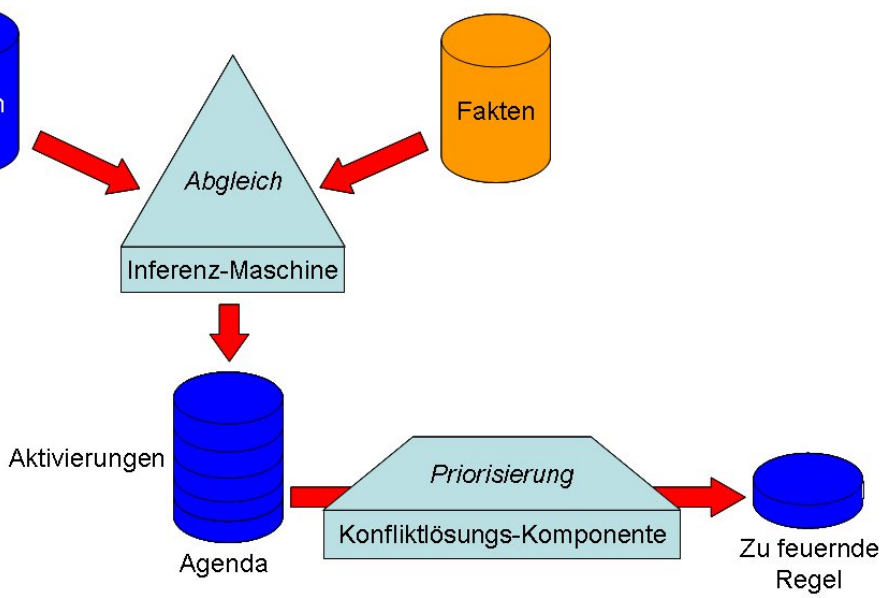


Abbildung 5.5: *Regelabgleich und Konfliktlösung in einem Expertensystem*

5.2.4 Die Benutzerschnittstelle

Die vierte Hauptkomponente eines Expertensystems bildet das Userinterface, dessen Aufgabe sowohl in der Darstellung und Erklärung der vom System abgeleiteten Schlussfolgerungen, als auch in der Bereitstellung einer Wissenserwerbskomponente liegt.

Das Userinterface eines Expertensystems kann allgemein in folgende Bestandteile gegliedert werden:

- **Erklärungskomponente**: *Wurde eine Aktivierung erzeugt, so muss der Nutzer darüber informiert werden. Die Erklärungskomponente führt dabei einerseits auf, welche Fakten aus der Wissensbasis zur Aktivierung der jeweiligen Regel geführt haben, während sie andererseits dafür zuständig ist, den Nutzer über die daraus folgenden Konklusionen in Kenntnis zu setzen.*

- **Interviewerkomponente**: *Je nach Anwendung kann es notwendig sein, weitere Daten vom Nutzer zu erfragen, die nicht von der Datennahmekomponente genommen werden können. Dazu stellt die Interviewerkomponente innerhalb des Userinterfaces spezielle Dialoge zur Verfügung.*

- **Wissenserwerbskomponente**: *Da jedes Expertensystem maßgeblich von Umfang und Qualität der vorhandenen Regelbasis abhängig ist, stellt die Wissenserwerbskomponente Möglichkeiten zur Verfügung, diese Regelbasis auszubauen und zu verfeinern. Zu unterscheiden ist dabei zwischen Systemen, in denen ein mit der Regelsyntax vertrauter Expertensystem-Administrator in Rücksprache mit den Fachexperten syntaxkonforme Regeln eingibt, und solchen, in denen ein Regeleditor direkt von den Fachexperten bedient werden kann (diese Situation ist in Abbildung 5.1 dargestellt).*

Weitergehende Details des Userinterfaces lassen sich nicht allgemeingültig für alle Expertensysteme definieren. Eine ausführliche Beschreibung des Userinterfaces für das im Rahmen dieser Arbeit erstellte Expertensystem “Pixel-Advisor” findet sich in Kapitel 8.

Kapitel 6

Anforderungen an ein DCS-Expertensystem

6.1 Motivation für ein DCS-Expertensystem

Die Motivation für den Einsatz eines Expertensystems für das DCS ist durch den hohen Anteil fachspezifischen Wissens gegeben, der eine Problemlösung für Nicht-Experten erschwert oder unmöglich macht. Andererseits sind große Bereiche des Fachwissens recht einfach über „wenn-dann"-Regeln zu realisieren, was den Einsatz eines regelbasierten Systems ermöglicht. Weiterhin ist der Großteil des Systems bereits automatisiert, so dass für das eigentliche DCS kein zusätzlicher Aufwand für die Datennahme entsteht.

Neben diesen grundsätzlichen Voraussetzungen für den Einsatz eines Expertensystems sind folgende wichtige Motive zu nennen:

- **Unterstützung der Nutzer & Entlastung der Experten**: *Hauptziel der Anstrengungen ist es, ein dediziertes Expertensystem so weit zu entwickeln, dass es im tatsächlichen Betrieb des Detektors hilft, die wenigen echten DCS-Experten zu entlasten, indem es dem normalen Schichtpersonal die selbstständige Lösung einfacher DCS-Probleme ermöglicht.*

- **Konservierung des Wissens**: *Ein weiterer wichtiger Motivationspunkt liegt in der kurzen durchschnittlichen Beschäftigungsdauer der Experten, die selten viele Jahre innerhalb des DCS tätig sind. Es ist daher von großem allgemeinen Interesse, dieses Fachwissen für die nachfolgenden DCS-Experten zu konservieren.*

Die folgenden Abschnitte 6.1.1 und 6.1.2 stellen die sich daraus ergebenden Anforderungen im Detail vor.

6.1.1 Was ein DCS-Expertensystem können MUSS

Folgende Eigenschaften sind als Anforderung für ein zu implementierendes Expertensystem anzusehen:

- **Keine Beeinträchtigung des DCS**: *Das DCS muss unabhängig vom Expertensystem weiterhin voll funktionsfähig bleiben.*
- **Bedienungsfreundlichkeit**: *Das Expertensystem soll einfach zu bedienen sein und die Arbeit des Schichtpersonals erleichtern.*
- **Leistungsfähigkeit**: *Das Expertensystem muss leistungsstark genug sein, um eine gewisse Zahl an einfachen Problemen zu lösen, um eine Grundakzeptanz bei den Nutzern zu erreichen.*
- **Einfacher Wissenserwerb**: *Der kritischste Punkt eines Expertensystems ist der Aufbau der Wissenserwerbskomponente. Ist diese zu kompliziert, beziehungsweise der Wissenserwerb zu aufwändig und zeitintensiv für den Experten, so wird dieser neue Fehlerbeschreibungen nicht in das System eingeben und das Expertensystem damit zum Scheitern verurteilen.*
- **Verständlichkeit**: *Die Fehlerbeschreibungen und Problemlösungsvorschläge sollten in einer möglichst einfach verständlichen Art und Weise, nach Möglichkeit in menschlicher Sprache, dargestellt werden, damit das Schichtpersonal den Entscheidungsprozess nachvollziehen und die Problemlösungen verstehen kann.*
- **Zurückhaltung**: *Lösungen sollten immer als „mögliche“ Lösungen mit einer gewissen Unsicherheit angegeben werden. Das System sollte immer davon ausgehen, dass nicht alle möglichen Ursachen eines Problems bekannt sind.*
- **Dynamik**: *Die Unsicherheitsfaktoren der verschiedenen Lösungen sollten dynamisch sein und sich den ändernden Verhältnissen während des Betriebs automatisch anpassen.*
- **Generalisierung**: *Neue Fehlerbeschreibungen sollten anhand von bereits aufgetretenen Einzelfehlern generalisiert werden können.*
- **Spezialisierung**: *Bereits bestehende Fehlerbeschreibungen sollten genauer spezifiziert werden können.*
- **Effizienz**: *Die Kernkomponenten des Expertensystems müssen auf einem einzigen Rechner schnell genug betrieben werden können.*

6.1.2 Was ein DCS-Expertensystem NICHT darf

Es ist besonders wichtig, bei der Entwicklung folgende Punkte sicherzustellen:

- **Autonomie beanspruchen**: *Das Expertensystem darf Probleme nicht ohne Nutzerinteraktion eigenständig lösen und muss mögliche automatische Problemlösungen vorher explizit und verständlich erklären.*
- **Stabilität beeinträchtigen**: *Das Expertensystem darf den stabilen Betrieb des DCS nicht durch zusätzliche Last, blockierende Abfragen oder sonstige Programmaspekte gefährden.*
- **Mensch oder Maschine gefährden**: *Das Expertensystem darf keine Lösungsvorschläge liefern, die Mensch oder Maschine gefährden könnten.*

6.2 Zusätzliche Rahmenbedingungen

Neben den im vorherigen Abschnitt ausgeführten (internen) Anforderungen an ein Expertensystem für das Detektorkontrollsystem des Pixeldetektors ergeben sich weitere externe Anforderungen an das System. Diese sind darauf begründet, dass im Rahmen der Doktorarbeit von Markus Mechtel, ebenfalls an der Universität Wuppertal, zur Zeit ein Expertensystem für das LHC-Grid[1] entwickelt wird.

Da große Teile der Anforderungen und der Infrastruktur in beiden Systemen sehr ähnlich sind, wurde vereinbart, eine gemeinsame Entwicklung zwei Einzellösungen vorzuziehen. Daraus ergeben sich weitere Vorteile, die in Kapitel 8 genauer beschrieben werden.

Da damit die zu nutzende Software auch für das Grid-Expertensystem "GridXP" eingesetzt werden soll, müssen einige zusätzliche Rahmenbedingungen eingehalten werden, die für das DCS-Expertensystem allein keine sehr hohe Priorität hätten:

- **Geschwindigkeit**: *Da das Expertensystem im Rahmen der Grid-Nutzung tausende "Jobs" überwachen soll, muss es möglichst schnell sein.*
- **Client/Server-Architektur**: *Das Expertensystem muss auf einer Client/Server-Architektur basieren, die den Einsatz mehrerer Clients auf entfernten Rechnern erlaubt.*

[1] Der Begriff des "Grids" ist schwierig zu definieren, was sich in einem Zitat von Ian Foster, von vielen als Vater des Grids bezeichnet, widerspiegelt: "The recent explosion of commercial and scientific interest in the Grid makes it timely to revisit the question: What is the Grid, anyway?"[21] Foster selbst liefert im selben Dokument eine mögliche Definition: "A Grid is a system that coordinates resources that are not subject to centralized control using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service." Eine genauere Beschreibung des LHC-Grid findet sich in [6] und [26]

- **Verschlüsselung**: *Die Kommunikation zwischen den Clients und dem Server muss verschlüsselt erfolgen, da angestrebte kommerzielle Nutzungen des Grid dies aus Gründen des Datenschutzes verlangen.*
- **Portabilität**: *Das Userinterface und der Server müssen auf Windows- und Linuxrechnern lauffähig sein.*

Neben diesen sich zusätzlich ergebenden, wichtigen Rahmenbedingungen muss das System im Detail weitere Forderungen erfüllen, die für einen alleinigen Betrieb im Rahmen des Pixel-DCS eventuell nicht notwendig gewesen wären. Da diese während der Software-Evaluierungsphase jedoch keine Rolle spielten, wird auf diese erst an den entsprechenden Stellen in Kapitel 8 hingewiesen.

6.3 Ausgangslage für die weitere Arbeit

Für die Implementation des Expertensystems, bietet sich der Einsatz bereits vorhandener Softwarekomponenten an. Diese sind in "Expertensystem-Shells" zu finden, die den Aufbau eines eigenen Expertensystem mit Hilfe vorgefertigter Komponenten ermöglichen.

6.3.1 Vorhandene Software

Während der Nachforschungen stellte sich heraus, dass derzeit in Industrie und Forschung im wesentlichen drei verschiedene Experten-Shells zum Einsatz kommen.

CLIPS[2] ist das älteste der drei Systeme und verfügt über eine sehr umfangreiche Dokumentation, sowie eine aktive Nutzergemeinde, was insbesondere bei anfänglichen Schwierigkeiten hilfreich ist. CLIPS unterstützt neben zahlreichen anderen Inferenzmechanismen auch den besonders schnellen und effizienten Rete-Algorithmus, arbeitet datengetrieben (vorwärtsverkettet), bietet eine auf LISP[3] basierende Logiksprache sowie umfangreiche Schnittstellen. CLIPS ist "Public Domain" und somit lizenskostenfrei.

JESS[4] ist die Java-basierte, weitgehende Reimplementation von CLIPS. Während CLIPS in C geschrieben ist und damit mehr Aufwand bei der Portierung von einem Betriebssystem auf ein anderes verursacht, läuft Java unter allen üblichen Betriebssystemen (Windows, Unix, Mac) ohne weitere Modifikationen. Da JESS nicht auf der neuesten CLIPS-Version basiert[5], die JESS-Community deutlich kleiner als die von CLIPS ist und

[2] **C L**anguage **I**ntegrated **P**roduction **S**ystem
[3] **Lis**t **P**rocessing
[4] **J**ava **E**xpert **S**ystem **S**hell
[5] JESS basiert auf einer CLIPS-Version aus dem Jahr 1995. Kurze Zeit später trennten sich die Entwicklungspfade.

JESS seitdem parallel zu CLIPS entwickelt wird, bietet JESS zudem nicht alle Features der letzten CLIPS-Versionen. JESS ist für den kommerziellen Gebrauch lizenzkostenpflichtig, nach Absprache für den universitären Gebrauch jedoch kostenlos.

"JBoss Rules" oder auch "Drools" ist ebenfalls eine auf CLIPS basierende und komplett in Java geschriebene Expertensystem-Shell. Im Gegensatz zu JESS ist JBoss Rules jedoch ein Open Source Projekt. Die hinter der JBoss-Produktfamilie stehende Firma RedHat[6] hat gerade im wissenschaftlichen Umfeld einen guten Ruf, was auch durch die Wahl von RedHat-Linux als Grundbaustein des am CERN und Fermilab genutzten "Scientific Linux" deutlich wird.

Die Entscheidung, welche der drei Expertensystemshells die bessere Wahl ist, war zu Beginn der Arbeiten am Pixel-DCS-Expertensystem anhand der vorliegenden Eckdaten nicht eindeutig zu fällen. Um die Eignung der Programme für das später zu erstellende DCS-Expertensystem auszuloten, wurden umfangreiche Tests durchgeführt, die in Abschnitt 7.3 genauer beschrieben werden.

6.3.2 Fehlende Komponenten

Unabhängig von der Wahl der oben skizzierten Expertensystem-Shells, müssen die einzelnen Softwarekomponenten modifiziert oder komplett bereitgestellt werden:

- **Die Regelbasis**: *Abhängig von den Möglichkeiten der Expertensystem-Shell muss eine geeignete Form der Regelbasis gefunden und an das System angebunden werden. Weiterhin muss die Regelbasis mit einem Initialsatz von Regeln gefüllt werden, um einen ersten Betrieb zu ermöglichen.*

- **Die Datennahmekomponente**: *Die PVSS-Daten müssen möglichst ohne großen Rechenaufwand und Zeitbedarf an das Expertensystem propagiert werden. Dabei ist zu beachten, dass eine vollständige Überwachung der mehr als 30000 DCS-Parameter (siehe Abbschnitt 4.2.1) von vornherein ausgeschlossen[7] ist. Daher muss ein geeigneter Triggermechanismus gefunden und implementiert werden, der die selektive Weitergabe aller notwendigen Prozessparameter in möglichst kurzer Zeit erlaubt.*

- **Die Nutzerschnittstelle**: *Das Userinterface sowohl für den normalen Nutzer, als auch für den Experten muss komplett neu entwickelt und programmiert werden. Dabei kommt der übersichtlichen und benutzerfreundlichen Gestaltung der Wissenserwerbskomponente eine zentrale Bedeutung zu.*

[6] RedHat Inc., Raleigh, California, USA
[7] Auf Grund des begrenzten Datendurchsatzes von DIM, siehe Abschnitt 4.2.5.1

6.3.3 Strategie

Ein Ziel dieser Arbeit ist die Erstellung eines funktionsfähigen Prototypen für ein DCS-Expertensystem anhand dessen Fähigkeiten dann über die Form des tatsächlichen Einsatzes im Bereich des Detektorkontrollsystems befunden werden kann. Dadurch kann verhindert werden, dass Sackgassen-Entwicklungen unnötige und zeitaufwändige Programmier- und Entwicklungsarbeiten nach sich ziehen, indem diese bereits im frühen Prototypen-Stadium erkannt werden. Um diesbezüglich fundierte Aussage treffen zu können, ist es allerdings notwendig, dass der Prototyp eine ausreichende "Seriennähe" aufweist, also über alle wesentlichen, in Abschnitt 6.1.1 erwähnten Merkmale verfügt und insbesondere gegen keinen der unter 6.1.2 aufgeführten Punkte verstößt.

Da am Anfang der Arbeit kaum absehbar war, welche technischen Schwierigkeiten die Entscheidung für oder gegen eine bestimmte Software beeinflussen, bzw. den Gesamtablauf hinauszögern würden, wurde die gesamte Entwicklung in mehrere überschaubare Schritte eingeteilt, auf deren Ergebnisse die folgenden Kapitel eingehen werden:

- **Evaluierungsphase (Kapitel 7):**
  Zuerst sollte ein sehr einfaches "Spielzeugexpertensystem" mit wenigen Regeln und manueller Dateneingabe für erste Tests implementiert werden. Hauptziel dieser Phase sollte es sein, erste Erfahrungen mit der generellen Problemstellung, der benötigten Software sowie der verwendeten Shell zu sammeln. Diese Phase sollte im Idealfall sowohl mit CLIPS, als auch mit JESS und JBoss Rules durchgeführt werden. Am Ende sollte eine Entscheidung zu Gunsten eines der drei Produkte getroffen werden, da eine weitergehende Parallelentwicklung zu zeitaufwändig gewesen wäre.

- **Designphase (Kapitel 8):**
  Nachdem das Spielzeugexpertensystem in den vorhergehenden Tests seine Funktionsfähigkeit unter Beweis gestellt hat, soll das System in der Designphase schrittweise an das finale System angepasst werden. Das eigentliche Design soll dabei am Anfang dieser Phase in Grundzügen festgelegt werden. Daraufhin mussten Datennahmekomponente, Userinterface und Regelbasis für das spätere Pixel-DCS-Expertensystem implementiert werden. Im weiteren Verlauf wurden dann alle Komponenten sukzessive angepasst, um alle Designkriterien (siehe 6.1) zu erfüllen.

- **Prototyp (Kapitel 8):**
  Aufbauend auf den bis dahin gemachten Erfahrungen sollte es möglich sein, einen komplett funktionsfähigen Prototypen zu programmieren. Dieser sollte bereits über alle später vorgesehenen Hauptfunktionen verfügen. Insbesondere das Userinterface dieses Prototypen musste ausgereift genug sein, um tatsächlich eingesetzt werden zu können.

- **Einsatzphase (Kapitel 9):**
  Eine erste Produktionsversion (Beta) des Expertensystems soll zeitgleich mit dem

ersten Betrieb des Pixeldetektors betriebsbereit sein. Es ist anzunehmen, dass in den ersten Tagen und Wochen des Detektorbetriebs eine Vielzahl an Fehlern auftritt, von denen wahrscheinlich viele auf Grund ihrer simplen Natur besonders geeignet für die Aufnahme in die Wissensdatenbank sind. Außerdem wird zu diesem Zeitpunkt ausreichend DCS-Manpower am CERN verfügbar sein, um das System zu beaufsichtigen. Die in dieser Zeit gewonnenen Erfahrungen sollten ausreichen, um am Ende der Phase den Erfolg des Projektes einschätzen zu können und über eventuelle Folgeprojekte nachzudenken.

Kapitel 7

Evaluation Expertensystem-Shells

Da das Pixel-DCS-Expertensystem eine komplette Neuentwicklung darstellt, sollen vorhandene Softwarekomponenten soweit wie möglich genutzt werden, um den Entwicklungsprozess so effektiv wie möglich zu gestalten. Die Eignung einiger dieser Werkzeuge war bereits am Anfang der vorliegenden Arbeit als erwiesen gegeben (7.1), während gerade die Auswahl der genutzten Expertensystem-Shell (die in Abschnitt 6.3.1 vorgestellt wurden) nur durch eine eingehende Erprobung der Kandidaten getroffen werden konnte (7.3).

7.1 Gemeinsame Werkzeuge

Das spätere Expertensystem wird aus den vier in Abschnitt 5.2 beschriebenen Komponenten Regelbasis, Datennahme, Inferenzmaschine und Userinterface bestehen. Jede dieser vier Komponenten kann prinzipiell in einer anderen Programmiersprache implementiert werden, was jedoch den Nachteil in sich birgt, dass die Schnittstellen zwischen den einzelnen Komponenten technisch aufwändig werden. Daher sollten die verwendeten Komponenten auf einen minimalen Satz unterschiedlicher Sprachen zurückgreifen.

7.1.1 Die Wahl von Java als Hauptprogrammiersprache

Da das Expertensystem (sowohl der Kern, als auch das Userinterface) gemäß den in Kapitel 6 aufgeführten Anforderungen auf Linux- und Windowsrechnern gleichermaßen eingesetzt werden soll, muss die genutzte Software möglichst einfach zwischen diesen Systemen zu portieren sein.

Die Programmiersprache Java bietet dabei als einzige Sprache die Möglichkeit, alle Komponenten des Expertensystems nativ zu unterstützen und ist von Natur aus porta-

bel[1]. Eine Implementation der einzelnen Komponenten in einer anderen Hochsprache wäre mit deutlich höherem Aufwand verbunden gewesen, da für keine der in Frage kommenden Sprachen wie C, C++, C# oder Python die in Kapitel 6 gegebenen Bedingungen gleichzeitig erfüllbar waren, ohne auf den Einsatz mindestens einer zweiten Hochsprache zurückzugreifen.

7.1.2 Die Entwicklungsumgebung Eclipse

Zu den Hilfsmitteln moderner Softwareprogrammierung gehört weiterhin die Nutzung einer leistungsfähigen IDE[2], die das Erstellen und Verwalten des notwendigen Quellcodes vereinfacht.

Nachdem die Wahl auf die Programmiersprache Java gefallen war, konnte der Kreis in Frage kommender IDEs stark eingeschränkt werden.

Die ursprünglich von IBM[3] entwickelte Entwicklungsumgebung "Eclipse" wurde im Jahr 2001 als Open Source Software freigegeben. Eclipse verfügt momentan über eine sehr aktive Entwicklergemeinde, die die IDE ständig um zusätzliche Plug-Ins erweitern.

Die große Nutzergemeinde, die Unterstützung moderner Programmiertechniken sowie die freie Verfügbarkeit qualifizieren Eclipse für alle im Rahmen der Implementation des Expertensystems anfallenden Arbeiten.

7.2 Das Spielzeugexpertensystem

Um fundierte Auswahlkriterien für die Wahl einer der drei in Frage kommenden Expertensystem-Shells zu liefern, musste zunächst ein Spielzeugmodell des späteren Expertensystems gebaut werden mit dem die Vor- und Nachteile der verschiedenen Lösungen verglichen werden konnten (siehe auch 6.3.3).

Um eine tiefgreifende Parallelentwicklung zu vermeiden, verfügt dieses Spielzeugexpertensystem lediglich über eine stark vereinfachte Datennahmekomponente, sowie ein rudimentäres Userinterface.

Das Hauptgewicht während der Entwicklung dieses Systems lag dabei in der Nachbildung eines Teils des Pixeldetektors und der Simulation aller relevanten Funktionen seines

[1] Java erreicht dies durch den Einsatz von "virtuellen Maschinen". Diese simulieren auf einem beliebigen Rechner eine virtuelle Java-Maschine, die die Besonderheiten des tatsächlichen Betriebssystems (Windows, Linux, Mac OS, etc.) vor dem Java-Applikationscode verbirgt. Einzige Voraussetzung für die Nutzung eines solchen Ansatzes liegt in der Verfügbarkeit der Java-Virtual-Machine auf den Zielrechnern. Am CERN ist eine jeweils aktuelle Version der Java-Laufzeitumgebung für alle Rechner verfügbar.

[2] **I**ntegrated **D**evelopment **E**nvironment

[3] **I**nternational **B**usiness **M**achines Corporation

FSM-Detektorastes. Abbildung 7.2 zeigt das Hauptübersichtspanel des simulierten Miniaturdetektors. Die gesamte Simulation wurde in PVSS geschrieben und genau wie das reale System über den PVSS-eigenen Datenpunktmechanismus realisiert.

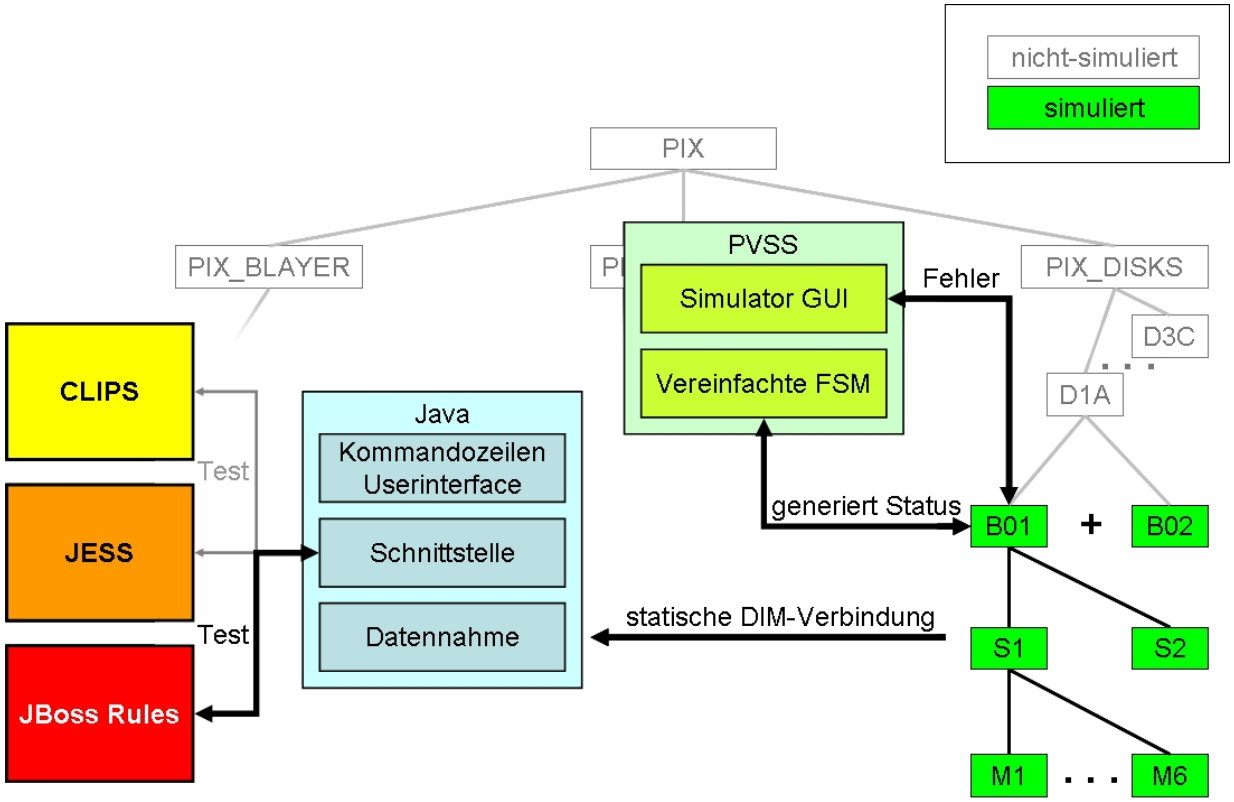


Abbildung 7.1: *Aufbau des Spielzeugexpertensystems*

Abbildung 7.1 zeigt den prinzipiellen Aufbau des zum Test der verschiedenen Expertensystem-Shells genutzten Spielzeugsystems. Grundsätzlich besteht das Modell dabei aus zwei Disc-PCCs, bzw. Bi-Sectors, sowie deren Sektoren und Modulen. Eine stark vereinfachte Nachbildung der FSM ist in der Lage, aus den gegebenen Modul- und Sektorwerten, Status zu generieren und diese, wie im realen Detektor, an die oberen Hierarchiestufen weiterzugeben. Das Userinterface des Detektor-Simulators erlaubt die gezielte Manipulation einzelner Werte wie der Niederspannungen VDD und VDDA oder die automatische Erzeugung von Zufallsfehlern durch das System selbst. Da nur die generellen Fähigkeiten der Expertensystem-Shells getestet werden sollen, verfügt das System für die meisten Werte nur über eine binäre “gut” oder “schlecht” Information. Lediglich die Temperaturen wurden der Realität entsprechend als Gleitkommazahlen implementiert, um die Eignung der verwendeten Regelsyntax für diesen Datentyp abschätzen zu können.

Während die Simulatorpanel in PVSS programmiert waren, sollte die restliche Software als Java-Applikation laufen und über den in Abschnitt 4.2.5.1 beschriebenen DIM-Mechanismus auf die PVSS-Datenpunkte des Simulators zugreifen. Im Gegensatz zur fina-

len Datennahmekomponente, die in Kapitel 8 noch näher beschrieben wird, kamen hier statisch im DIM-Server veröffentlichte Information-Services zum Einsatz. Mit anderen Worten musste sich die Java-Applikation nicht darum kümmern, die PVSS-Datenpunkte über den DIM-Mechanismus zu publizieren, sondern konnte sich sofort mit diesen verbinden.

Dabei wurde bereits ein auch im späteren System eingesetzter Mechanismus etabliert: der Prototyp verband sich zunächst nur mit den Status der obersten Hierarchiestufe (also den beiden Bi-Sektoren). Spezielle "Kontrollregeln" innerhalb der Regeldatenbank reagierten dann bei einer Status-Wertänderung, indem sie zusätzliche Informationen über die darunter liegende Ebene anforderten, bzw. wieder freigaben.

Auf unterster Ebene wurden dann die ermittelten Temperaturen (Gleitkommazahlen) und alle anderen Werte (boolsch) von "Werteregeln" erfasst.

Das generelle Vorgehen war dabei das manuelle oder automatische Generieren eines Fehlers im Simulator, woraufhin der Expertensystem-Prototyp eine entsprechende Ursachenerklärung, wie etwa "Bi-Sector 2 im Status WARNING, weil Temperatur von Modul S2_M3 zu hoch!" geben sollte.

Diese vergleichsweise einfach klingende Aufgabe stellte allerdings durchaus hohe Anforderungen an die Systeme, da bereits eine vollständig geschlossene Datenübertragung zwischen Simulator und Prototyp gegeben sein musste.

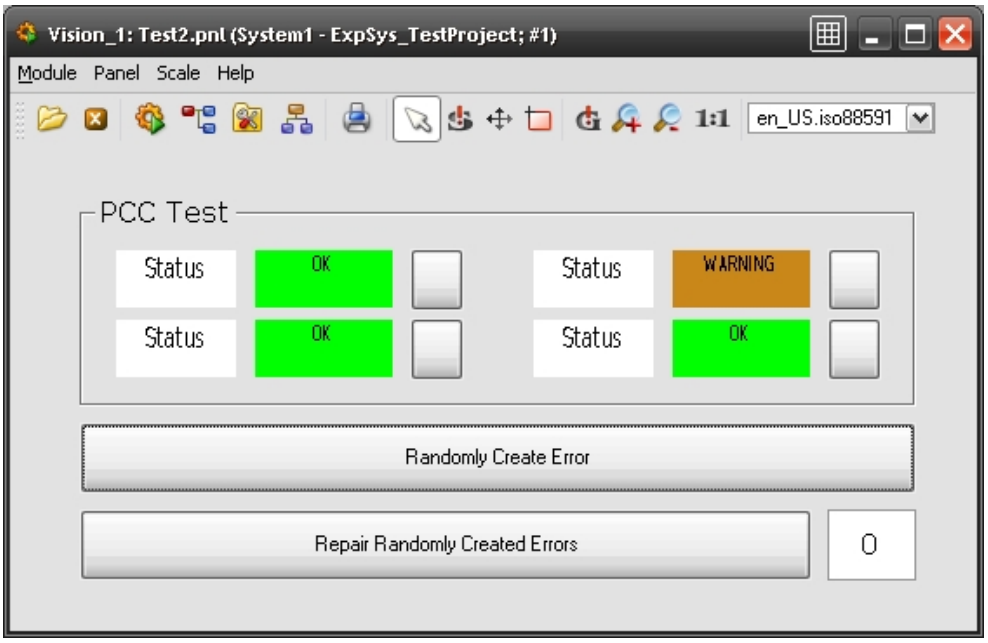


Abbildung 7.2: *Übersichtspanel des Spielzeugexpertensystems*

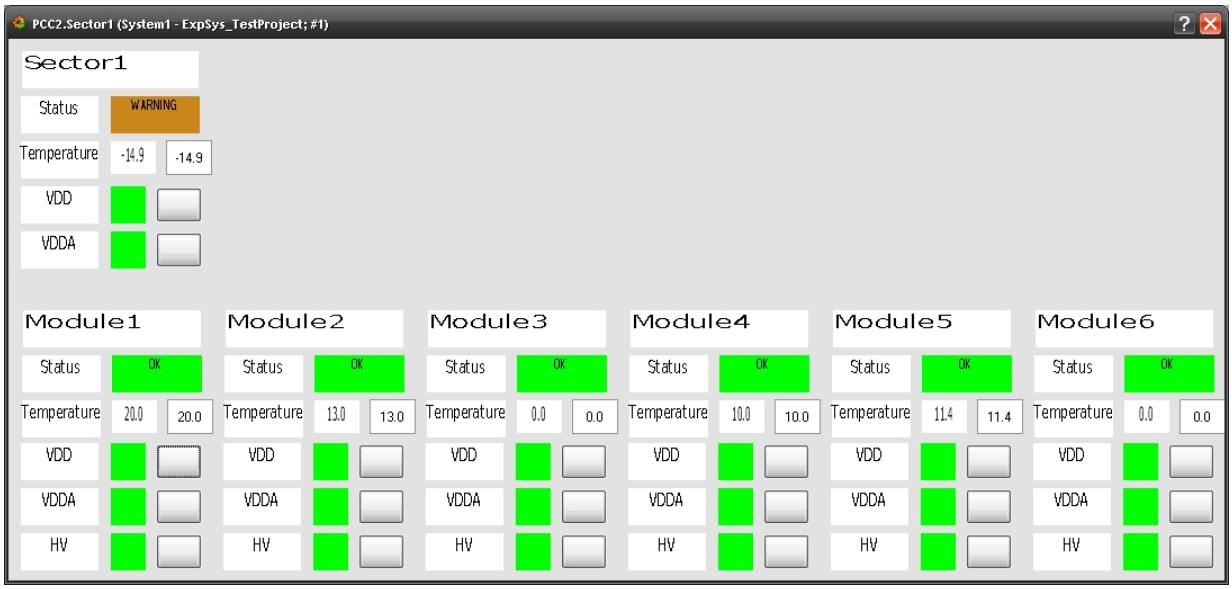


Abbildung 7.3: *Detailpanel eines Spielzeugsektors*

7.3 Test der Kandidaten

Um die Eignung der einzelnen Kandidaten zu testen, wurde versucht, diese nacheinander in den Aufbau des Spielzeugexpertensystems einzubauen. Die dabei gemachten und für die letztliche Entscheidung ausschlaggebenden Erfahrungen werden in den folgenden Abschnitten aufgeführt.

7.3.1 CLIPS

Da die beiden anderen Kandidaten auf CLIPS aufbauen und jeweils nur eine Teilmenge der CLIPS-Fähigkeiten bieten, wurde das System zunächst mit CLIPS getestet.

Wie sich schnell zeigte, war die Wahl des C-basierten CLIPS im Zusammenhang mit der Java-Programmiersprache ungünstig. Die bestehende Java-Portierung "JCLIPS" verfügte nur über einen Teil der ursprünglichen CLIPS-Funktionen und erwies sich darüberhinaus als teilweise fehlerhaft und schlecht dokumentiert (CLIPS selbst ist hingegen recht gut dokumentiert). Die alternative Anbindung von CLIPS durch JNI[4] kam hingegen auf Grund des zusätzlichen Aufwandes nicht in Frage, da dies im wesentlichen eine Neuimplementation des unzureichenden JCLIPS bedeutet hätte.

Auf Grund der vorhandenen Alternativen wurde die Arbeit mit CLIPS daher frühzeitig abgebrochen.

[4] **J**ava **N**ative **I**nterface. Erlaubt die direkte Ausführung nativer Anwendungen wie etwa C oder C++ Funktionen

7.3.2 JESS

Im Gegensatz zu CLIPS handelt es sich bei JESS um eine direkte Java-Implementation, so dass hier keine Schwierigkeiten bei der Ankopplung an die Java-Applikation zu erwarten waren.

Tatsächlich zeigte sich, dass JESS eine robuste JAVA-Version der CLIPS-Shell bot, die relativ leicht an die Datennahme angeschlossen und so mit dem Simulator verbunden werden konnte.

Die Dokumentation war dabei allerdings deutlich weniger präzise, als die von CLIPS, was zweifellos dazu führte, dass nicht alle Möglichkeiten der Software innerhalb der Testphase ausgelotet werden konnten.

Als gewichtigster Nachteil der JESS-Shell erwies sich die altmodische und unflexible, von CLIPS geerbte Regelsyntax. Es ist in JESS weder möglich, Regeln in natürlicher Sprache zu verfassen, noch ist die JESS-eigene Syntax intuitiv und durch einen späteren Nicht-Experten leicht zu verstehen.

```
(defrule Temperature-Rule
    "A simple temperature rule"
    (Module ?Name ?Temperature)
    (test (not (-88.8 ?Temperature)))
    (or (test (< -20.0 ?Temperature)) (test (> 40.0 ?Temperature)))
=>
    (printout t "Reason for problem is a temperature of "
    ?Temperature "°C on Module " ?Name)
)
```

Abbildung 7.4: *Beispielregel in JESS. Die Syntax ist identisch mit der von CLIPS und beruht weitgehend auf der ehemaligen LISP-Syntax.*

Abbildung 7.4 zeigt eine einfache Regel in JESS-Syntax. Zunächst werden Felder für die Temperatur und den Namen eines jeden Detektormoduls angelegt. Danach wird die Modultemperatur mit dem symbolischen Wert “-88.8” verglichen, der im Pixel-DCS für ein nicht verbundenes Kabel steht und schließlich wird überprüft, ob sich die Temperatur im erlaubten Bereich befindet.

Die zum Zeitpunkt der Testphase aktuelle Version 7.0 von JESS zeigte ausgeprägte Schwächen in der Nutzung der durch die Datennahmekomponente bereitgestellten Daten innerhalb der Regeln. Ein direkter Zugriff auf einzelne Java-Objekte war lediglich nach der “Bindung” von Regelvariablen an die einzelnen Member eines Objektes möglich.

Weiterhin war die Dokumentation der Regelsprache selbst zwar sehr gut, dafür wurden die bereitgestellten Java-Funktionen größtenteils lediglich computergeneriert und wenig einsteigerfreundlich dokumentiert.

Im Gegensatz zum Open-Source-Gemeinschaftsprojekt CLIPS wird das kommerzielle, für akademische Nutzung kostenlose JESS lediglich von einer einzigen universitären Gruppe betreut. Sollte diese Gruppe die Arbeit an der Software beenden, wäre eine Zukunft dieser Software ungewiss.

7.3.3 JBoss Rules

Auch bei JBoss Rules liegt eine direkte Java-Implementation einer CLIPS-basierten Expertensystem-Shell vor. Im Gegensatz zu JESS ist das auch als Drools bezeichnete JBoss Rules allerdings eine komplette Neuauflage einer Expertensystem-Shell, die sich lediglich an den von CLIPS eingebürgerten Grundideen orientiert.

Ein großer Vorteil von Drools ist die einfache und vollständige Integration der Expertensystem-Shell in die Eclipse-IDE (siehe Abschnitt 7.1.2) mit Hilfe eines Plug-Ins. Dieses bietet einen Drools-Regeleditor, eine Ansicht des Rete-Baumes sowie spezielle Ansichten zur Verwaltung der verschiedenen Komponenten, die die Programmierung erleichtern.

Die Regelsyntax von JBoss Rules ist dabei grundsätzlich von CLIPS/JESS verschieden. Während sowohl CLIPS als auch JESS auf die mittlerweile sehr alte LISP-Syntax zurückgreifen, nutzt Drools eine stark an herkömmliche Programmier-Hochsprachen angelehnte Syntax. Der Aufbau zusammengesetzter logischer Ausdrücke, die Behandlung numerischer Werte, sowie die Zuweisung von Variablen werden damit deutlich vereinfacht.

Der aus Sicht des Programmierers größte Vorteil der Drools-Shell liegt in der Fähigkeit, direkt aus der Regelmaschine auf die Java-Objekte einer Applikation zugreifen zu können. Das “Binden” jedes einzelnen Attributs eines eventuell sehr komplexen Java-Objekts entfällt. Diese wichtige und vornehmlich im Bedingungsteil einer Regel genutzte Eigenschaft wird sogar soweit erweitert, dass der direkte Aufruf von Java-Code aus dem Anweisungsteil einer Regel möglich ist.

Der wohl auffälligste Unterschied zu den beiden bisher angesprochenen Shells ist die Unterstützung natürlichsprachlicher Regeln durch den DSL[5]-Mechanismus. Obwohl einfach aufgebaut, erlaubt es dieser Mechanismus doch, die Lesbarkeit einer Regel für Nutzer, die über kein tiefgreifendes Wissen der eigentlichen Regelsyntax verfügen, stark zu verbessern. Abbildung 7.5 zeigt zwei Realisationen der bereits in Abschnitt 7.3.2 verwendeten Regel in JBoss Rules. Dabei wurde einmal die eigentliche Drools-Syntax und einmal eine natürlichsprachliche DSL genutzt.

Die untere in Abbildung 7.5 gezeigte Regel ist dabei ohne große Vorkenntnisse über den Aufbau des Expertensystems oder dessen Regelsyntax zu verstehen. Das Anlegen einer DSL geschieht über eine einfache Textdatei, die in einer Spalte die natürlichsprachlichen

[5] **D**omain **S**pecific **L**anguage

```
rule "Temperature Rule"
    //A simple temperture rule.
    when
        Module($name : name,
        $temperature : temperature != -88.8 &&
        (temperature< 20.0 || temperature > 40.0))
    then
>       System.out.println("Reason for problem is a temperature of "
>       $temperature "°C on Module " $name);
end

rule "Temperature Rule"
    //A simple temperture rule with DSL.
    when
        There is a Module temperature problem
    then
        Print reason for problem
end
```

Abbildung 7.5: *Die einfache Temperaturregel in JBoss Rules. Oben ohne und unten mit Nutzung der Domain Specific Language*

Ausdrücke wie beispielsweise “There is a Module temperature problem” und in einer zweiten Spalte deren Drools-Übersetzung enthält.

In der Abbildung ebenfalls sichtbar ist der Einsatz von nativem Java-Code im Anweisungsteil der Regel durch den Funktionsaufruf “System.out.println(“. . . ”). Der Aufruf nativer Java-Funktionen aus der Regelmaschine heraus ist bei anderen Expertensystem-Shells nicht oder nur auf Umwegen möglich.

Als Teil der von RedHat gepflegten JBoss-Produktfamilie verfügt Drools zudem über die im Vergleich aktivste Nutzer- und Entwicklergemeinde. Vorteilhaft machte sich dies insbesondere in der Antwortzeit auf im Entwicklerforum gestellte Fragen bemerkbar, die hier deutlich kürzer als bei der Konkurrenz war. Der Einsatz mehrerer Vollzeitkräfte seitens RedHat garantiert zudem die Stabilität und zukünftige Weiterentwicklung der Software, während der Open-Source-Gedanke die Erweiterung durch externe Plug-Ins ermöglicht.

Die Anbindung an den Simulator innerhalb des Prototypen-Expertensystems war weitestgehend problemlos möglich, so dass vielversprechende Resultate schneller als mit den beiden anderen Shells erzielt werden konnten.

Der einzige gravierende Nachteil, der bisher im Zusammenhang mit der Nutzung von Drools sichtbar wurde, ist die nur unzureichende Anbindung an eine SQL[6]-basierte Regeldatenbank. Zwar unterstützt die Software als einzige der Kandidaten überhaupt die Möglichkeit, die Regeln an Stelle einer Textdatei direkt aus einer Datenbank zu lesen, jedoch geht bei diesem Schritt der wichtige DSL-Mechanismus verloren. Dass die anderen

[6] **S**tructured **Q**uery **L**anguage

Shell	CLIPS	JESS	JBoss Rules
Open Source	ja	nein	ja
Programmiersprache	C	Java	Java
Qualität der Dokumentation	+/-[a]	+/-	+
Aktivität der Community	-	- -	+
Zukunft der Software	+/-	-	+
Integration in IDE	- -	+/-	++
Rete-Algorithmus	ja	ja	ja
Fuzzy-Logik	ja	nein	(nein[b])
Natürlichsprachliche Regeln	nein	nein	ja
Regelsyntax	+/-	+/-	+
Direkter Zugriff auf externe Variablen	nein	(ja[c])	ja
Direkter Zugriff auf Java-Funktionen	nein	nein	ja
Einsatz moderner Softwaretechnik	–	+/-	++
Nutzung einer SQL-Regeldatenbank	nein	nein	(nein[d])

Tabelle 7.1: *Leistungsmerkmale der verschiedenen Expertensystem-Shells*

[a] Bewertung: ++ = sehr gut, + = gut, +/- = mittelmäßig, - = schlecht, - - = sehr schlecht

[b] Eine Arbeitsgruppe der Fachhochschule Steinfurt um Prof. Wulff plant eine Fuzzy-Erweiterung von JBoss Rules, die erstmals unsicheres Schließen ermöglichen wird.

[c] Erfordert direkte Bindung der Regelvariablen an die einzelnen Objekt-Membervariablen

[d] Nutzung einer SQL-Datenbank als Regelbasis wird unterstützt. Dabei geht bisher allerdings die Möglichkeit der Nutzung einer DSL verloren.

Tools über gar keine direkte Anbindungsmöglichkeit an Datenbanken verfügen, die Anbindung an eine solche, im Rahmen dieser Arbeit aus Zeitgründen ohnehin unmöglich gewesen wäre und die Unterstützung von DSL-Datenbankregeln in zukünftigen JBoss Rules Versionen sicherlich hinzugefügt wird, führt dazu, dass dieser Punkt weitgehend vernachlässigt werden konnte. Da zumindest das Pixeldetektor-Expertensystem prinzipiell nicht auf eine Datenbank angewiesen ist, spielt er für die weiteren Betrachtungen keine Rolle.

7.4 Gegenüberstellung und Auswahl

Nachdem die drei Expertensystem-Shells in den vorangegangenen Abschnitten mit Hilfe des Spielzeugexpertensystems auf ihre Tauglichkeit für das Pixeldetektor-Expertensystem geprüft wurden, liefert Tabelle 7.1 eine Gegenüberstellung der wichtigsten Leistungsmerkmale.

Nach den in der obigen Tabelle zusammengefassten Erfahrungen dieses Kapitels und der Absprache mit Markus Mechtel, dem Entwickler des parallel zu implementierenden GridXP, fiel die Entscheidung für die genutzte Shell auf JBoss Rules.

Kapitel 8

Pixel-Advisor: Ein Expertensystem für DCS

Der Schwerpunkt der vorliegenden Arbeit liegt in der Konzeption und Ausarbeitung eines voll funktionsfähigen Expertensystems für das Pixeldetektor-DCS. Mit dem "Pixel-Advisor" wurde eine entsprechende Software entwickelt und implementiert, die im Rahmen dieses Kapitels detailliert beschrieben werden soll.

8.1 Aufgabenteilung

Wie bereits in Abschnitt 6.2 erwähnt, ist neben dem Pixel-Advisor noch ein weiteres Expertensystem, das GridXP, entwickelt worden. Um den Entwicklungsaufwand in beiden Projekten so gering wie möglich zu halten und die spätere Wartbarkeit der Software zu optimieren, sind Teile der in den folgenden Abschnitten erwähnten Komponenten in Absprache für beide Systeme gemeinsam entworfen und implementiert worden. Dabei handelt es sich um den Userinterface-Client sowie die eigentliche Kernkomponente der Software, die die Inferenzmaschine und den Server beinhaltet. Diese gemeinsame Komponentensammlung, die von beiden Systemen genutzt wird, wird intern mit dem Namen "UnifiedXP" bezeichnet. Somit bestehen sowohl der Pixel-Advisor als auch das GridXP aus den UnifiedXP-Komponenten sowie ihrer jeweils eigenen Regelbasis und Datennahmekomponente.

Da Regelbasen und Datennahmekomponenten keine nutzbaren Gemeinsamkeiten aufweisen, wurden diese für GridXP und Pixel-Advisor getrennt entwickelt.

Abgrenzung

Das **UnifiedXP**-Projekt stellt die Kernkomponenten eines auf JBoss Rules basierenden Expertensystems zur Verfügung. Da UnifiedXP selbst weder über Datennahme- noch Regelbasis-Komponenten verfügt, ist es kein eigenständiges Expertensystem.
Die beiden Expertensysteme **GridXP** und **Pixel-Advisor** bauen auf UnifiedXP auf, verfügen aber über ihre eigenen Datennahmekomponenten und Regelbasen.

Der Aufbau der beiden auf den UnifiedXP-Komponenten basierenden Expertensysteme ist Abbildung 8.1 zu entnehmen.

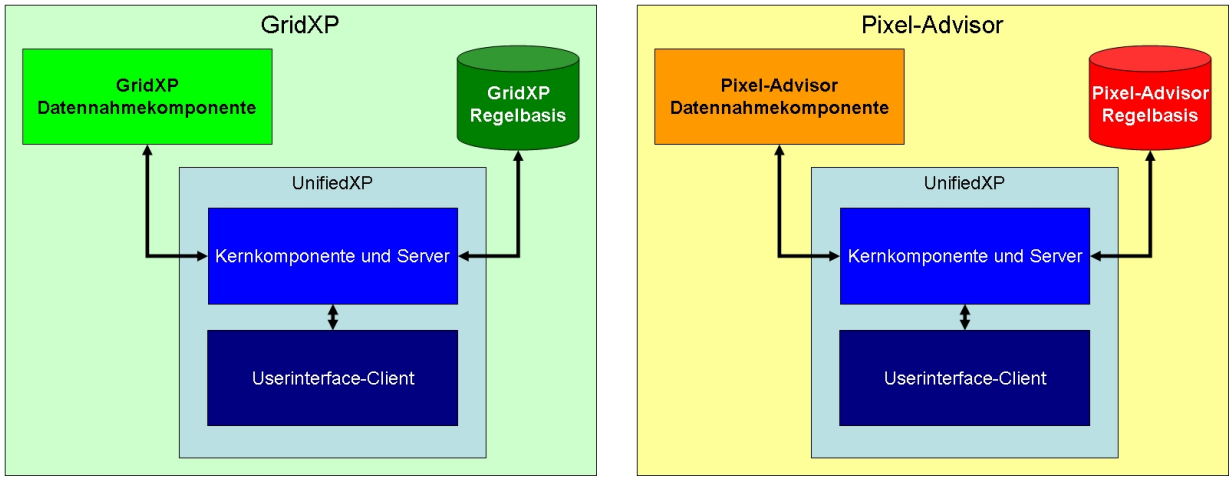


Abbildung 8.1: *Komponentenvergleich von GridXP und Pixel-Advisor. Das aus Kernkomponente und Userinterface bestehende UnifiedXP wird von beiden System genutzt.*

Die Auswertung möglicher Gemeinsamkeiten führte zur Vergabe zweier Informatik-Bachelorarbeiten, in deren Rahmen die gemeinsamen Komponenten implementiert wurden.

Währenddessen konzentrierten sich die Anstrengungen dieser Arbeit auf die Weiterentwicklung der Komponenten, die vom Pixel-Advisor allein genutzt werden: der Regelbasis und der Datennahmekomponente.

Zwei Studenten der Informatik der Fachhochschule Steinfurt, Dennis Huning und Frank Iker, übernahmen im Rahmen ihrer Bachelorarbeiten technische Aspekte der Implementation der gemeinsamen Komponenten.

Für die Arbeiten an dem gemeinsamen Userinterface war dabei Dennis Huning verantwortlich, während das insbesondere vom GridXP geforderte Client-Server-Konzept von Frank Iker implementiert wurde. Die Zusammenführung der bisher getrennten Code-Basen von GridXP und Pixel-Advisor auf ein gemeinsames Software-Fundament war Teil beider Bachelorarbeiten.

Eine genaue Auflistung der Aufgabenbereiche und Verantwortlichkeiten der (in alphabetischer Reihenfolge aufgelisteten) Beteiligten ist durch Tabelle 8.1 gegeben.

	Tobias Henß	Dennis Huning	Frank Iker	Markus Mechtel
Pixel_Advisor				
Datennahme	✓			
Regelbasis	✓			
UnifiedXP				
Vereinheitlichung	✓	✓	✓	✓
Kernkomponente	✓	✓	✓	✓
Userinterface	✓	✓		
Client/Server			✓	✓
GridXP				
Datennahme				✓
Regelbasis				✓

Tabelle 8.1: *Aufgabenteilung innerhalb von GridXP, Pixel-Advisor und UnifiedXP*

In Bezug auf das Userinterface und die Client-Server-Kommunikation hat die Aufgabenteilung, im Rahmen dieser Arbeit, eine Konzentration auf die Designentscheidungen und konzeptionellen Probleme ermöglicht. Detaillierte Informationen zur technischen Umsetzung der beiden gemeinsamen Komponenten finden sich in den Bachelorarbeiten [29] und [30].

Auch für die im Rahmen dieser Arbeit selbstgeschriebenen Komponenten, die Regelbasis und die Datennahmekomponente, sollen hier keine technischen Details im Sinne einer Programmdokumentation angeführt, sondern lediglich die Kernkonzepte und wesentlichen Designentscheidungen der einzelnen Komponenten erläutert werden.

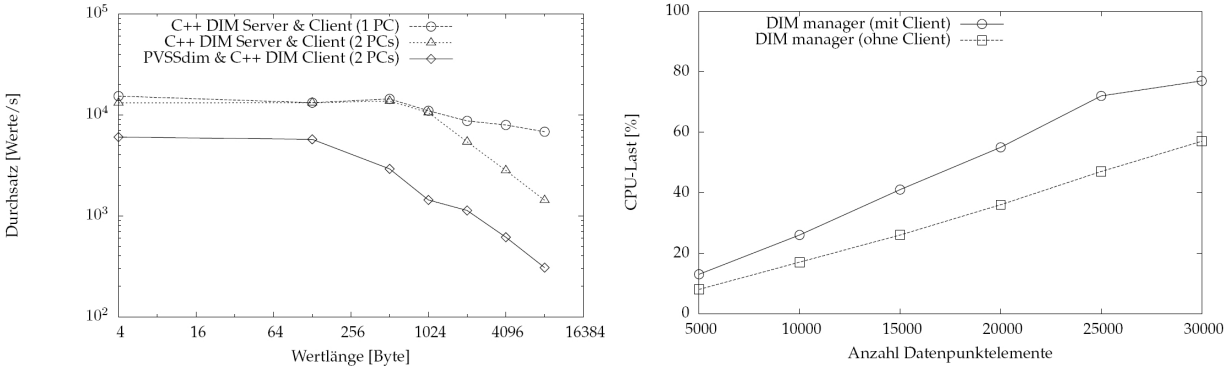
8.2 Die Datennahmekomponente

Eine der wesentlichen Herausforderungen bei der Implementation des Pixel-Advisors bestand in der Anbindung des Expertensystems an die PVSS-basierte Kontrollsystemsoftware des Pixel-DCS. Innerhalb der DCS-Software liegen die Prozessparameter wie Temperaturen, Spannungen und Ströme in Form der bereits in Kapitel 4, Abschnitt 4.2.1 erwähnten Datenpunkte vor. Bei diesen handelt es sich nicht um ausschließlich im Arbeitsspeicher des Rechners abgelegte Variablen, sondern um Werte, die ebenfalls direkt auf die Festplatte geschrieben werden. Während dieser Mechanismus für ein Prozessleitsystem wünschenswerte

Vorteile mit sich bringt[1], macht er es kompliziert, extern auf diese Variablen zuzugreifen, da nicht einfach die Speicheradressen an externe Programme übergeben werden können, ohne die Konsistenz der Daten zu gefährden.

Da mit dem im Abschnitt 4.2.5 erwähnten DIM bereits eine vom CERN offiziell unterstützte Möglichkeit existierte, PVSS-Datenpunkte an externe Programme anzubinden, nutzt der Pixel-Advisor DIM als grundlegenden Datennahmemechanismus.

Das Detektorkontrollsystem des ATLAS-Pixeldetektors verwaltet mehr als 30000 Prozessparameter (siehe Abbildung 4.5), die sich im Schnitt etwa alle 5 Sekunden ändern können[2]. Die durchschnittliche Datenrate des Pixel-DCS beläuft sich also auf etwa 6000 Wertänderungen pro Sekunde.



(a) Datendurchsatz in Werte/s in Abhängigkeit der Wertlänge in Byte. Für Pixel ist der Bereich einfacher Datentypen bis 16 Byte relevant.

(b) CPU-Last gegen Anzahl der in 5 s-Intervallen aktualisierten Datenpunktelemente, sowohl mit als auch ohne den Abruf der Werte durch einen Client.

Abbildung 8.2: *Durchsatz und CPU-Last der DIM-Kommunikation gemäß [41].*

Dem gegenüber steht der begrenzte Datendurchsatz von DIM, da dieses nie für die Übermittlung von großen Datenmengen konzipiert wurde. Ursprüngliche Angaben (2004) der DIM-Entwickler am CERN beliefen sich auf etwa 1000 Wertänderungen pro Sekunde, die über längere Zeiträume stabil weitergegeben werden konnten. Die Leistungsfähigkeit des DIM-Protokolls hängt dabei maßgeblich von der auf dem Übertragungsweg eingesetzten Hardware ab. Da diese von Jahr zu Jahr schneller wird, verwundert es nicht weiter, dass neuere Studien[41] zeigen, dass deutlich höhere Datenmengen via DIM übertragbar sind (siehe auch Abbildung 8.2). Allerdings zeigt sich auch, dass die Performance weiterhin stark von externen Faktoren, wie etwa der Anzahl der mit einem Server verbundenen Clients oder der Größe der zu übertragenden Werte in Bytes abhängt. Offensichtlich ist die theoretisch benötigte Datenrate von 6000 Werten/s am Rande der Machbarkeit, bringt den eingesetzten PVSS-DIM-Manager allerdings bereits in einen Bereich sehr hoher Grundlast.

[1] bei Absturz des Systems befinden sich keine wichtigen Daten im Arbeitsspeicher, die dann verloren gehen könnten

[2] Diese maximale Auslesefrequenz ist im Wesentlichen durch die Geschwindigkeit der eingesetzten Analog-Digital-Wandler begrenzt.

Alle DIM-nutzenden Subsysteme (FSM, DDC und Pixel-Advisor) sind daher angehalten, die DIM-Datenrate deutlich unter der maximal möglichen Last zu halten. Dies ist insbesondere wichtig, da die einzelnen FSM-Prozesse über den DIM-Mechanismus untereinander Daten austauschen und diese Kommunikation unter allen Umständen vor Störeinflüssen geschützt werden muss. Eine pauschale Verbindung zu allen DCS-Prozessparametern des Pixeldetektors scheidet damit aus.

Da Stabilität und Leistungsfähigkeit des DCS-Systems durch den Pixel-Advisor unter keinen Umständen gefährdet werden dürfen (siehe Anforderungen 6.1.2), wurden Maßnahmen getroffen, um eine zu hohe, durch den Pixel-Advisor bedingte DIM-Last zu vermeiden.

Dazu wurden zwei wesentliche Vorkehrungen getroffen:

- **Statische Maximallast:** *Um den DIM-Mechanismus nicht zu überfordern, sieht der Pixel-Advisor die Begrenzung der maximalen Zahl an DIM-Items vor, mit der er sich gleichzeitig verbinden kann. Dieses Limit kann über eine Konfigurationsdatei eingestellt werden. Ist die angegebene Maximalzahl an DIM-Items des Pixel-Advisors erreicht, werden keine weiteren DIM-Verbindungen mehr hinzugefügt. Da DIM auch noch von anderen Programmen genutzt wird (beispielsweise von DDC) liegt das Limit unterhalb der theoretisch möglichen DIM-Datenrate, momentan voreingestellt bei* 200 *DIM-Items.*
- **Dynamische Lastanpassung:** *Der Pixel-Advisor versucht zu jedem gegebenen Zeitpunkt nur solche Werte über DIM anzufordern, die auch wirklich zur Untersuchung des vorliegenden Problems benötigt werden. Unproblematische Detektorteile werden dabei sofort wieder freigegeben, um die DIM-Last zu reduzieren.*

Da der zweite Mechanismus im Detail aufwändig ist, soll er im folgenden Abschnitt genauer beschrieben werden.

8.2.1 Dynamische Lastanpassung

Das Prinzip der dynamischen Lastanpassung der Datennahmekomponente ist an die DCS-FSM gekoppelt, da diese alle Prozessparameter des Pixeldetektor-Kontrollsystems überwacht und daraus State/Status-Paare bildet (siehe Abschnitt 4.2.4). Diese State/Status-Paare enthalten eine komprimierte Zustandsinformation über den Pixeldetektor. Weil diese Zustände darüberhinaus der Detektorgeographie (Abschnitt 3.1, sowie Abbildung 8.3) folgend nach oben propagiert werden, lässt sich die FSM nutzen, um die Pixel-Advisor-Datennahme zu steuern.

Dazu wird der Status einer FSM-Control- bzw. Device-Unit überwacht. Nimmt dieser einen problematischen Wert ein (“WARNING”, “ERROR” oder “FATAL”), so weist die Datennahmekomponente den PVSS-DIM-Server über einen dafür reservierten

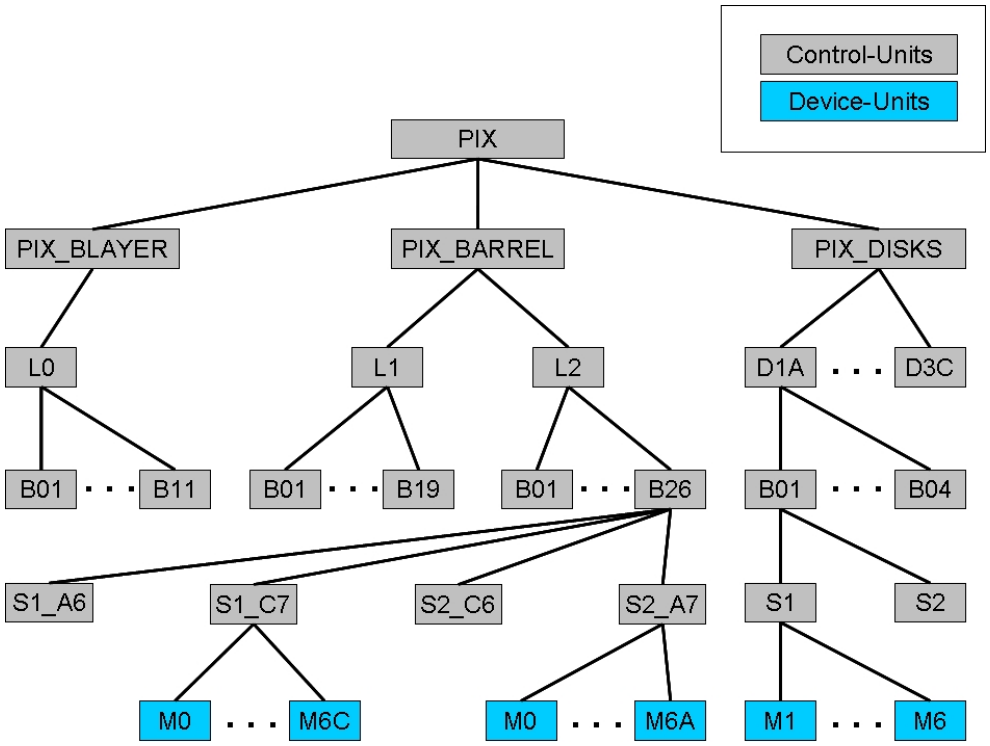


Abbildung 8.3: *Hierarchie der Pixeldetektor-DCS-FSM mit Control- und Device-Units. Das Schema ist vereinfacht, so sind aus Gründen der Übersichtlichkeit beispielsweise nicht alle Device-Units, wie die Opto-Boards etc. eingezeichnet.*

Expertensystem-RPC an, die Werte der Kinder dieser Kontroll-/Geräteeinheit ebenfalls zu veröffentlichen. Je nachdem ob es sich bei den Kindern selbst um Control/Device-Units oder um Werte handelt, bestehen diese aus Status oder Prozessparametern. Sobald eine Kontroll- oder Deviceeinheit aus einem problematischen Status in einen unproblematischen "OK"-Status wechselt, werden die Werte sämtlicher Kinder freigegeben. Da FSM-Status immer von unten nach oben propagiert werden, ist gewährleistet, dass auch die Kindeskinder eines beliebigen Knotens bei einem solchen Wechsel freigegeben werden.

Im Normalzustand des Detektors verbindet sich die Datennahmekomponente daher lediglich mit den obersten Hierarchiestufen der Detektor-FSM: "PIX_BARREL", "PIX_BLAYER" und "PIX_DISKS". Wird nun innerhalb einer Detektorpartition, wie in Abbildung 8.4 angedeutet, ein kritischer Prozessparameter registriert, wird dieser zu den obersten Hierarchieebenen propagiert. Dort wird die Spur, im Beispiel das Auftreten einer gefährlich hohen Temperatur[3], bis zu ihrer Quelle verfolgt.

Die dazu notwendige Kontrolllogik ist dabei in Form von Regeln implementiert, was in Abschnitt 8.3 genauer erläuterte Vorteile mit sich bringt, auf die hier zunächst noch

[3]Das Beispiel ist etwas übertrieben, da ab einer Temperatur von 40 °C bereits ein Interlocksignal des Interlocksystems zu einer Notabschaltung des betroffenen Half-Staves geführt hätte.

nicht eingegangen werden soll. Im Falle eines kritischen Status weisen diese “Datenfluss”-Kontrollregeln die Datennahmekomponente an, die Verbindung zu den direkten Kindern der betroffenen FSM-Einheit herzustellen. Im Beispiel löst der “ERROR”-Status der PIX_BARREL-Partition die Nachfrage einer Verbindung zu den Layer-Partitions “L1” und “L2” aus.

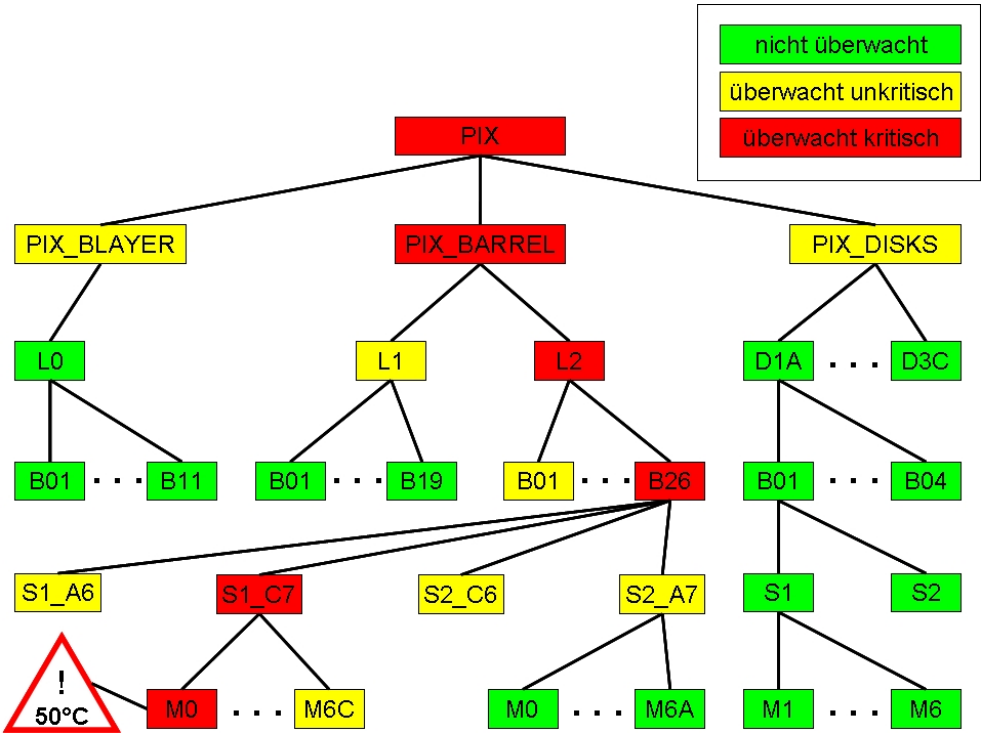


Abbildung 8.4: *Dynamische Lastanpassung des Pixel-Advisors mit Hilfe der DCS-FSM-Zustandsinformationen.*

Da nicht davon ausgegangen werden kann, dass der DIM-Server die betreffenden Werte bereits als Information-Services bereitstellt, sendet die Datennahmekomponente zunächst ein “PUBLISH”-Kommando an den bereits erwähnten Expertensystem-Kommandodatenpunkt (siehe Abschnitt 4.2.5.1 und Abbildung 4.10). Dieser wird daraufhin von dem ebenfalls bereits in Abschnitt 4.2.5 beschriebenen PVSS-Skript gelesen. Sollte der angeforderte Datenpunkt noch nicht im DIM-Server bereitgestellt, also veröffentlicht worden sein, so wird dies nachgeholt. Zuletzt wird der Name des DIM-Services, unter dem der Client den angeforderten Datenpunkt im DIM-Name-Server erreichen kann, an die Datennahmekomponente zurückgegeben. Sollte dieser Datenpunkt speziell für den Pixel-Advisor veröffentlicht worden sein (also für den Fall, dass der Datenpunkt noch nicht veröffentlicht war), so wird der DIM-Service mit einem “xpt_”-Präfix versehen, welches später noch von Bedeutung ist. Die Datennahme verbindet sich dann umgehend mit diesem Information-Service.

Da im Beispiel der Zustand des "L2" ebenfalls kritisch ist, werden auch dessen DCS-FSM-Kinder über oben beschriebenen Mechanismus angefordert. Dieser Vorgang wird vollständig durch die Datenfluss-Kontrollregeln gesteuert und wiederholt sich solange, bis die ursächliche Device-Unit lokalisiert wurde.

Sobald dies erreicht ist, werden deren Prozessparameter über weitere Kontrollregeln angefordert und mit Hilfe, ebenfalls in Abschnitt 8.3 genauer beschriebener, "Werte"-Regeln überprüft.

Nachdem das Problem gelöst und die Temperatur des im Beispiel betroffenen Moduls "L2_B26_S1_C7_M0" wieder auf ein normales Niveau gebracht wurde, registriert die FSM dies und ändert den Status des Moduls auf einen unkritischen Wert. Das wiederum löst die Ausführung einer weiteren Kontrollregel aus, die die Prozessparameter des Moduls freigibt[4].

Da die Datenpunkte der Modul-Prozessparameter nun nicht mehr benötigt werden, wird die bestehende DIM-Verbindung zu den entsprechenden DIM-Services zunächst getrennt. In Analogie zum Verbindungsvorgang wird dazu ein "UNPUBLISH"-Befehl von der Datennahmekomponente an den DIM-Server geschickt, um die DIM-Items im Sinne einer Lastreduzierung freizugeben. Sollte der Service-Name des freizugebenden Datenpunktes kein "xpt_"-Präfix enthalten, wird dieser Wert nicht exklusiv für das Expertensystem veröffentlicht und die Aufforderung an den DIM-Server, die Bereitstellung des Services einzustellen, wird ignoriert. Wie im Falle der Verbindungsherstellung läuft auch dieser Mechanismus intern über die Kommunikation von Datennahmekomponente und DDC-Skript durch den Expertensystem-Kommandodatenpunkt ab.

Nachdem der Status des Moduls von der FSM für unkritisch befunden wurde und auch keines der anderen Half-Stave-Module einen problematischen Zustand meldet, wechselt der FSM-Status des ehemals betroffenen Half-Staves "S1_C7" auf einen unkritischen Wert und der Freigabevorgang wiederholt sich auf der darüberliegenden Ebene. Nach zwei weiteren Iterationen befindet sich der Detektor wieder in seinem Grundzustand und der Pixel-Advisor ist lediglich mit den drei anfänglich erwähnten Hauptpartitionen statisch verbunden (die durch den Pixel-Advisor verursachte DIM-Last liegt also bei lediglich drei DIM-Items).

Sollte im Laufe des oben beschriebenen, dynamischen Verbindungsvorgangs das global gesetzte Limit für die Anzahl der gleichzeitig vom Pixel-Advisor angeforderten DIM-Items

[4]Bedingt durch die Ausleserate der eingesetzten ADCs kann es allerdings bis zu 6.5 s dauern, bis die Wertänderung von der FSM erkannt wird. Weiterhin benötigen die daraufhin ablaufenden Skripte, je nach Anzahl der freizugebenden DIM-Items, weitere Sekunden, so dass durchaus bis zu 10 s zwischen Lösung des Problems und erfolgter Freigabe der DIM-Parameter vergehen können. Im (bisher nicht beobachteten) Fall ungünstig "auf der Schwelle" oszillierender Werte, ist es dadurch bedingt durchaus möglich, dass das Expertensystem nicht genug Zeit hat, die Daten zu holen, bevor diese wieder freigegeben werden. Dieses Verhalten ließe sich durch die in Abschnitt 9.2.3 erwähnte Nutzung historischer Daten vermeiden, was jedoch eine Anbindung an die Conditions Datenbank erfordern würde, die im Rahmen dieser Arbeit nicht vorgenommen werden konnte

erreicht werden, so unterbleibt die Verbindung zu den unteren Hierarchieebenen an entsprechender Stelle. Während dadurch ein ordnungsgemäßer Betrieb des DIM-Mechanismus und damit des Gesamtdetektors gewährleistet wird, birgt diese Sicherheitsmaßnahme aber auch die Gefahr eines "blockierten" Expertensystems. Ein solches Szenario wäre das gleichzeitige[5] Auftreten vieler kritischer Prozessparameter, wie es beispielsweise ein Ausfall des Kühlsystems bewirken könnte. Allerdings sind auch solche Situation noch durch den Einsatz eines regelbasierten Systems zu bewältigen, da sich im Allgemeinen ein bestimmtes Muster an als kritisch gemeldeten Detektorpartitionen bestimmten potentiellen Ursachen zuordnen lassen sollte. In einem solchen Fall könnte das System, "unter Verdacht" einer möglichen Ursache, gezielt Teile der verbundenen Daten freigeben und die freigewordenen Lastkapazität dazu nutzen, die Vermutung zu überprüfen. Da solche Regeln offensichtlich sehr komplex sind, wurden zum Zeitpunkt dieser Arbeit jedoch keine entsprechenden Maßnahmen implementiert.

8.3 Die Regelbasis

Die von UnifiedXP und damit vom Pixel-Advisor genutzte Inferenzmaschine JBoss Rules (siehe Kapitel 7) verwendet, wie alle vergleichbaren Systeme, eine Regelbasis in Form einer oder mehrerer Textdateien. Innerhalb dieser Dateien können beliebig viele Regeln definiert werden, deren Struktur durch die Regelsyntax der Inferenzmaschine vorgegeben ist.

Abbildung 8.5 zeigt eine einfache Regel, die gemäß der Drools-Regelsyntax erstellt wurde. In JBoss Rules wird eine Regeldefinition durch das Schlüsselwort "rule" begonnen und durch "end" beendet. Jede Regel besitzt einen eindeutigen Namen, der direkt nach dem rule-Schlüsselwort in Anführungszeichen gesetzt wird. Die Prämisse, bzw. der Bedingungsteil einer Regel wird dabei durch das Schlüsselwort "when" eingeleitet und durch das Wort "then" beendet, das den Anweisungsteil, auch Konklusion genannt, eröffnet. Wie in vielen modernen Programmiersprachen, können Kommentare durch "//" an beliebiger Stelle und in beliebiger Zahl angebracht werden, um die Lesbarkeit der Regeln zu erhöhen.

Einer der wichtigsten Vorteile, die Drools gegenüber seinen direkten, in Kapitel 7 erwähnten Konkurrenzprodukten aufweist, ist der Einsatz einer Domain Specific Language. Obwohl diese Eigenschaft bereits in Abschnitt 7.3.3 anhand eines Beispieles erklärt wurde, sei hier ein weiteres Mal darauf hingewiesen, wie stark sich die Lesbarkeit einer Regel durch den Einsatz der DSL verbessert. Zum direkten Vergleich zeigt Abbildung 8.6 die obige Regel ohne den Einsatz von DSL. Zwar ist diese Form der Regelsyntax für den

[5]Relevant ist dabei die Zeit, die das Expertensystem braucht, um von der Feststellung eines kritischen Status auf oberster Ebene zum problematischen Prozessparameter zu navigieren. Da im Laufe dieses Vorgangs viele aufwändige und synchronisierte Schritte auf Seiten des Expertensystems und des DCS zu absolvieren sind, kann dieser Vorgang mehrere Sekunden in Anspruch nehmen. Gleichzeitigkeit bedeutet in diesem Sinne daher das Auftreten eines zweiten Problems, bevor der kritische Parameter des ersten Problems lokalisiert werden konnte.

```
rule "Top Level Stati Investigation Rule"
    //"Rule to evaluate the top level stati."
    when
        There is a Sector status problem
    then
        //get additional data...
        Get Sector data
        Report status problem
end
```

Abbildung 8.5: *Beispiel einer einfachen Regel*

```
rule "Top Level Stati Investigation Rule"
    //"Rule to evaluate the top level stati."
    when
>       CommonDataObject($name : name matches ".*Sector..Status", type == Type.STRING,
                         $stringValue : stringValue != "OK" && stringValue != "UNKNOWN")
    then
        //get additional data...
>       insertLogical(new CommonDataObject("GetROUData", Type.STRING, $name));
>       System.out.println("Status problem on " + $name + "! Status is: " + $stringValue);
end
```

Abbildung 8.6: *Beispiel der Regel aus Abbildung 8.5, jedoch ohne Einsatz einer DSL*

Software-Experten immer noch lesbar, für an der Entwicklung des Expertensystems unbeteiligte Personen jedoch sicherlich sehr gewöhnungsbedürftig.

Um den DSL-Mechanismus nutzen zu können, werden zusätzlich zu den eigentlichen Regeldateien noch weitere Dateien benötigt, die die Übersetzungstabelle zwischen den “Schnipseln” oder “Snippets” natürlicher Sprache und den entsprechenden Code-Blöcken enthalten.

Ein wesentliches Ziel des Designs des UnifiedXP und damit auch des Pixel-Advisors ist daher naheliegenderweise die strikte Trennung der Bearbeitung dieser beiden Arten von Regeldateien. Während die DSL-Übersetzungsdateien ausschließlich von Expertennutzern bearbeitet werden dürfen, sollen die eigentlichen Regeldateien selbst auch durch Nutzer modifiziert werden können, deren Erfahrungsschatz sich auf den Umgang mit dem Detektorkontrollsystem selbst beschränkt.

Da ein regelbasiertes System mit der Verlässlichkeit seiner Regeln steht und fällt, ist die Bewahrung der Integrität der Regelbasis eine der Hauptverantwortlichkeiten der Software. Das Editieren bestehender oder Anlegen neuer Regeln ist daher innerhalb des UnifiedXP als zweistufiger Prozess implementiert.

Zunächst soll die Benutzerschnittstelle (Abschnitt 8.4) dem Nutzer die Möglichkeit bieten, Regeln auf einfache Art und Weise zu erstellen oder zu ändern, indem dieser aus

einem Katalog bestehender natürlichsprachlicher Ausdrücke (welche in den DSL-Dateien definiert sind) Regelfragmente zu kompletten Regeln zusammenklicken kann. Nachdem die Regel vom User fertiggestellt wurde, wird diese an die in Abschnitt 8.5 behandelte Serverkomponente übermittelt. Dort werden die geänderten oder neuen Regeln jedoch zunächst lediglich in einem lokalen Cache als "Regelvorschläge" gespeichert und vorerst NICHT in die genutzte Regelbasis übernommen (siehe Abbildung 8.7). Sobald sich einer der Software-Administratoren, bzw. Fachexperten (ebenfalls über das Client-Userinterface) mit den entsprechenden Berechtigungen anmeldet, hat dieser die Möglichkeit, die angesammelten Regelvorschläge zu sichten. Dabei obliegt es der Verantwortung des Administrators, eventuell nach weiterer Rücksprache mit den Detektorexperten, die vorhandenen Vorschläge zu korrigieren und in die tatsächliche (nur auf dem Server vorhandene) Regelbasis zu übernehmen.

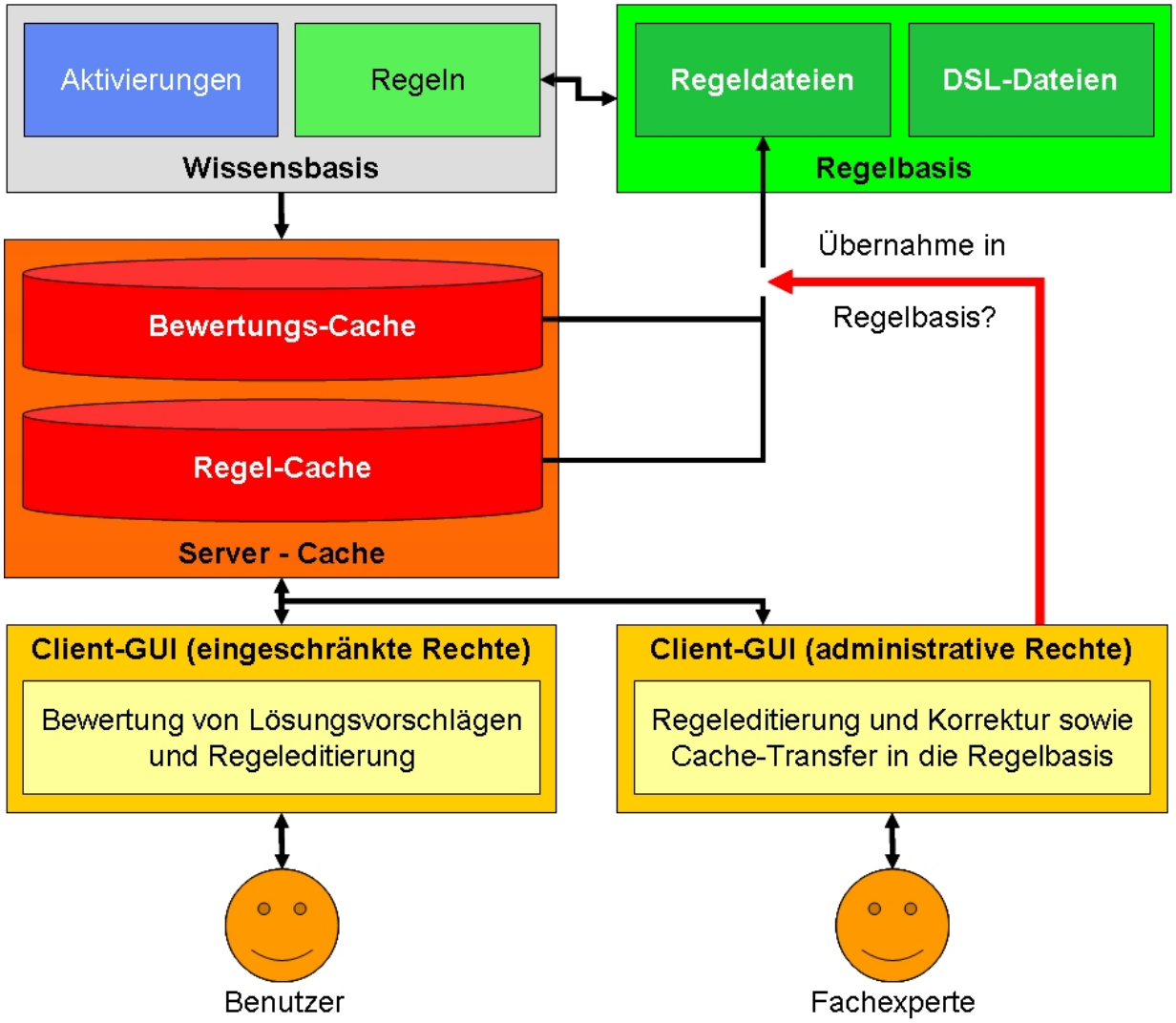


Abbildung 8.7: *Funktionsprinzip des Server-Caches.*

Durch diesen Sicherheitsmechanismus wird verhindert, dass ein unerfahrener Nutzer versehentlich irreführende oder gar falsche Regeln in das System eingeben kann.

Ein weiterer Vorteil dieses Vorgehens lag darin, dass Nutzer selbst dann Regelvorschläge unterbreiten können, wenn der oben erwähnte Regeleditor noch gar nicht oder nur rudimentär existiert, wie es zu Beginn der Arbeiten am Pixel-Advisor der Fall war. Obwohl unklar war, wann genau das Userinterface um einen voll funktionsfähigen Regeleditor erweitert werden konnte, konnten Nutzer so bereits Regelvorschläge in Form normaler, natürlichsprachlicher Sätze in das System eingeben, die dann natürlich von einem der Administratoren in eine syntaxkonforme Form gebracht werden müssen[6].

Da die verwendete Regelsyntax, insbesondere durch die Möglichkeit der direkten Nutzung von Java-Quellcode, sehr mächtig und flexibel ist, und sich komplexe logische Zusammenhänge darüberhinaus einfacher in Form einer Regel als durch verschachtelte "if-else"-Anweisungen realisieren lassen, kam frühzeitig die Idee auf, auch Teile der Programmsteuerung des Pixel-Advisors in Regelform zu implementieren. Konkret betrifft dies die bereits im Zusammenhang mit der dynamischen Lastanpassung in Abschnitt 8.2.1 erwähnten "Datenfluss"-Kontrollregeln. Diese steuern den Datenfluss der DIM-Datennahmekomponente und lassen sich sehr gut in Form von Regeln formulieren, wie das Beispiel in Abbildung 8.5 zeigt.

Um die Übersichtlichkeit der Regeldateien zu gewährleisten, werden diese Kontrollregeln und ihre dazugehörigen DSL-Snippets in einem von den eigentlichen "Werteregeln" getrennten Satz von Regeldateien verwaltet.

Neben der einfacheren Darstellung durch Regeln bringt die Auslagerung der internen Kontrolllogik noch einen weiteren, wesentlich gewichtigeren Vorteil mit sich: da der Pixel-Advisor damit nur noch aus einer sehr allgemein formulierten DIM-Datennahmekomponente, dem ebenfalls nicht Pixel-spezifischen UnifiedXP und der Regelbasis besteht, lässt sich die Software prinzipiell auch für andere Detektoren nutzen.

An dieser Stelle soll nochmals die Bedeutung des DIM-Protokolls hervorgehoben werden. Da DIM sich sowohl an die DCS- als auch an die DAQ-Software ankoppeln lässt und diese ATLAS-(teilweise sogar LHC-)weit auf derselben Softwarearchitektur basieren, ist das Pixel-Advisor-Expertensystem durch den Austausch einer einzigen Komponente, der Regelbasis, für einen anderen Einsatzzweck (DAQ) oder gar einen anderen Detektor nutzbar. Tabelle 8.2 macht diese Zusammenhänge anhand der verschiedenen Komponenten der Pixel-Advisor-Software deutlich.

Einige der vorhandenen Regeln werden vermutlich nach einer gewissen Zeit an Wichtigkeit verlieren, da die ursächlichen Probleme endgültig behoben werden konnten. So ist anzunehmen, dass nach den vorgesehenen Zugangsperioden vermehrt Probleme mit der

[6]Da vor Einsatz des Pixel-Advisors am CERN ein voll funktionsfähiger Regeleditor in das Userinterface integriert werden konnte, wurde diese Vorsichtsmaßnahme allerdings nie in Anspruch genommen.

Komponente	**geeignet für**
Pixel-Advisor-Regelbasis	Pixeldetektor DCS
Pixel-Advisor-Datennahmekomponente	alle DIM-basierten Softwarepakete des CERN, wie DCS und DAQ der meisten LHC-Subdetektoren
UnifiedXP-Userinterface	alle UnifiedXP-basierten Expertensysteme, also neben dem möglichen Einsatz in anderen LHC-Subdetektoren auch für GridXP.
UnifiedXP-Server	alle UnifiedXP-basierten Expertensysteme, alle LHC-Subdetektoren sowie GridXP.

Tabelle 8.2: *Einsatzgebiete der Komponenten des Pixel-Advisor*

Verkabelung des Detektors auftreten, welche sich nach einigen Tagen oder Wochen des Betriebs stabilisiert haben sollten.

Weiterhin werden im Allgemeinen mehrere Regeln für einen gegebenen Datensatz aktiviert und damit dem Nutzer präsentiert. Allerdings muss davon ausgegangen werden, dass bestimmte Regeln häufiger zutreffen als andere, von der Inferenzmaschine als gleichwertig angesehene Regeln.

Um dieser Problematik Rechnung zu tragen, bietet der Regelmechanismus des Pixel-Advisors die Möglichkeit, die Regelwahrscheinlichkeiten durch den Nutzer beeinflussen zu lassen. Dazu wird dem Nutzer im Userinterface, das in Abschnitt 8.4 genauer beschrieben wird, die Möglichkeit geboten, die einzelnen vorgeschlagenen Regeln zu bewerten. Wird eine Regel als zutreffend bewertet, wird dabei ein interner Zähler, "Salience" genannt, inkrementiert, während die Ablehnung einer Regel den Zähler dekrementiert.

In Analogie zur Vorgehensweise im Fall der Regelvorschläge werden auch diese Regel-Bewertungen nicht direkt in die Regelbasis übernommen, sondern zunächst in einem Zwischenspeicher des Servers abgelegt. Sobald ein Administrator das nächste Mal die Regelbasis erneuert, werden die kumulierten Zählerwerte in die Regelbasis übernommen. Je nachdem wie oft eine Regel dabei von eventuell unterschiedlichen Userinterface-Clients bestätigt oder als nicht-zutreffend deklariert wurde, resultiert daraus eine positive oder negative Änderung des bestehenden Salience-Zählers.

Sobald die Inferenzmaschine zu einem gegebenen Problem einen Satz an Aktivierungen erzeugt, berechnet ein austauschbarer Algorithmus[7] aus den aktuellen Salience-Werten der aktivierten Regeln eine "Wahrscheinlichkeit" für das Zutreffen einer jeden aktivierten Regel. Um nicht den Eindruck eines allwissenden oder unfehlbaren Expertensystems zu

[7]Damit wird es ermöglicht, die Salience-Werte unterschiedlich zu gewichten; etwa logarithmisch, linear...

vermitteln, besteht die Möglichkeit diese Wahrscheinlichkeiten zu begrenzen, damit Ihre Summe immer kleiner als 100% ist[8].

8.4 Die Benutzerschnittstelle

Das User-Interface des Pixel-Advisor ist als eigenständig lauffähige Java-Applikation die Schnittstelle zwischen dem eigentlichen Expertensystem und dem Nutzer. Dabei erfüllt die graphische Oberfläche folgende wichtige Eigenschaften:

- **Anzeige der Werte-Regel-Aktivierungen:** *Während die Kontrollregeln sofort nach ihrer Aktivierung ausgeführt werden, bestimmt der Nutzer des Expertensystems im Falle der Aktivierung einer Werte-Regel selbst, ob er die vorgeschlagene Problemlösung überprüft oder nicht. Das User-Interface übernimmt dabei die Aufgabe, dem Nutzer alle zu einem Problem gehörenden Problemlösungen anzuzeigen.*
- **Bereitstellung eines Regeleditors:** *Damit bestehende Regeln modifiziert und neue Regeln hinzugefügt werden können, stellt das User-Interface einen Regeleditor bereit, der dem normalen Nutzer oder dem Expertensystem-Administrator den Zugriff auf die Regeldatenbank erlaubt.*

Dementsprechend ist das User-Interface in zwei Ansichten unterteilt, die in den nächsten Abschnitten behandelt werden sollen.

8.4.1 Die Aktivierungsansicht

Sobald die Datennahmekomponente des Pixel-Advisors Fakten liefert, die zur Aktivierung einer Werte-Regel führen, soll dem Nutzer der in der entsprechenden Werte-Regel vorhandene Lösungsvorschlag präsentiert werden. Verkompliziert wird dies durch die hohe Anzahl an FSM-Device-Units, die alle als potentielle Quellen problematischer Fakten fungieren, im Zusammenspiel mit der Möglichkeit einer Vielzahl gleichzeitig auftretender Probleme. In einem solchen Szenario würde sich eine einfache und unstrukturierte Aktivierungsliste sehr schnell mit einer schwer überschaubaren Anzahl an Aktivierungen füllen und damit zusätzlich zur Verschärfung einer ohnehin kritischen Situation sorgen.

Daher stellt der Pixel-Advisor nicht alle Aktivierungen des Pixeldetektors gleichzeitig in einer einzelnen Liste dar, sondern gruppiert Aktivierungen gemäß ihres Auftretens in der Detektorhierarchie.

[8]Dies wird auch durch die Namensgebung Pixel-Advisor = Pixel-BERATER unterstrichen, die den fehlbaren und insbesondere keine Allwissenheit vorgaukelnden Charakter des Expertensystems betonen soll.

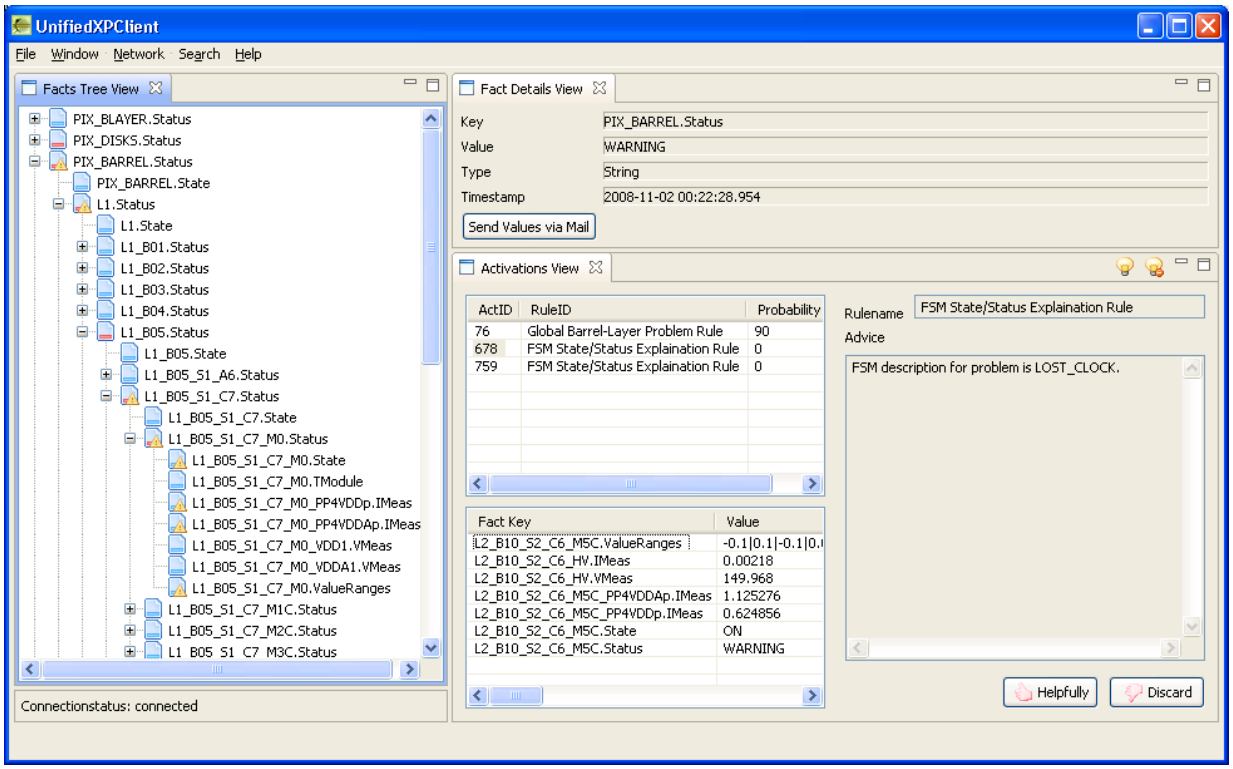


Abbildung 8.8: *Aktivierungsansicht des UnifiedXP-Client. Dieser Screenshot wurde auf einem Windows-XP-Rechner gemacht. Das Userinterface ist durch den Einsatz von Java unabhängig vom genutzten Betriebssystem lauffähig.*

Abbildung 8.8 zeigt einen Screenshot der Aktivierungsansicht der Userinterfaces. Auf der linken Seite des Fensters befindet sich eine, der FSM-Hierarchie folgende, Baumansicht der geographischen Detektorstruktur. Durch Auswahl eines Knotens innerhalb dieser "Facts Tree View", wird das durch diesen Knoten repräsentierte Datum in der "Fact Details View" angezeigt, die sich auf dem Beispielbild rechts-oben befindet. In diesem Faktenfenster werden Name, Wert, Datentyp und Zeitstempel der letzten Wertänderung des ausgewählten Faktums angezeigt, also Status für Device- und Control-Units der FSM und Temperaturen, Spannungen oder Ströme für die Prozessparameter des Pixeldetektors. Die rechts-unten sichtbare "Activations View" beinhaltet die eigentliche Aktivierungsansicht der Benutzeroberfläche. Dabei werden in diesem Fenster lediglich die Aktivierungen angezeigt, die von einem Zweig oder Blatt des ausgewählten Knotens generiert wurden. Mit anderen Worten also nur solche Aktivierungen, die den FSM-Status des gewählten Knotens beeinflussen. Eine Auswahl des "PIX_BARREL"-Knotens zeigt dementsprechend die Aktivierungen aller Barrel-Module an, während auf der Ebene eines Half-Staves nur noch die Aktivierungen der zugehörigen 6 oder 7 Module angezeigt werden.

Da der Pixel-Advisor die UnifiedXP-Client-Komponente als Userinterface nutzt, besitzt das Userinterface selbst keinerlei Informationen über den Aufbau des Detektors. Dies wä-

re ein Widerspruch zur allgemeinen Nutzbarkeit des UnifiedXP-Clients und wurde daher vermieden. Statt dessen wird die komplette Baumstruktur durch Informationen aufgebaut, die direkt aus den Fakten ausgelesen werden können. Dazu wird eine im Abschnitt 8.5 näher beschriebene Containerklasse für die Fakten genutzt, die sowohl von GridXP, als auch vom Pixel-Advisor genutzt wird und die sicherstellt, dass jeder Knoten seine jeweiligen Eltern und Kinder kennt. Da die dazu notwendigen Informationen ausschließlich aus der Regelbasis gewonnen werden, genügt im Falle des Pixel-Advisors ein Austausch der Regeldateien, um das Expertensystem für einen anderen (LHC-)Detektor nutzen zu können, dessen Struktur dann ebenfalls durch den Baum dargestellt werden würde.

Von der Existenz einer Aktivierung abhängige Icons in der Baumstruktur erlauben dem Nutzer die schnelle Lokalisierung von problematischen Detektorpartitionen und die einfache Navigation zur betroffenen Detektoreinheit (siehe Abbildung 8.8)[9].

Nachdem der Nutzer den untersten und damit ursächlichen Knoten innerhalb der Detektorgeographie angewählt hat, werden nur noch die Aktivierungen angezeigt, die mit dem auf dieser Ebene vorhandenen Problem zusammenhängen. Dabei werden die Aktivierungen in einer Liste innerhalb der "Activations View" angezeigt, die den Namen der zugehörigen Regel sowie eine Regelwahrscheinlichkeit beinhaltet, nach der die Aktivierungen in der Liste auch geordnet werden.

Diese Wahrscheinlichkeit wird über einen definierbaren Algorithmus und den Salience-Werten der vorhandenen Aktivierungen bestimmt. Eine Regel mit hoher Salience wird dabei als wahrscheinlicher als eine mit geringer Salience angegeben.

Bei Auswahl einer der Aktivierungen werden die auslösenden Fakten[10] in der darunterliegenden Faktenliste der Activations View aufgelistet.

Da die Aktivierungsliste lediglich die Regelnamen und Wahrscheinlichkeiten enthält, wird bei Auswahl einer Aktivierung deren Lösungsvorschlag in einem zusätzlichen "Advice"-Feld angegeben.

Unterhalb dieses Feldes kann der Nutzer dem System eine Rückmeldung darüber geben, wie hilfreich der aktuelle Vorschlag bei der Lösung des Problems war. Wurde der Vorschlag als hilfreich empfunden, so führt dies zu einer Erhöhung des Salience-Wertes der dazugehörigen Regel, wohingegen ein Ablehnen des Vorschlags (weil dieser das Problem nicht lösen konnte) zu einer Verringerung der Regel-Salience führt.

Damit Änderungen an den Salience-Werten der Regeln jedoch Einfluss auf die Gewichtung der Regelwahrscheinlichkeiten haben, müssen diese Änderungen in die Regelbasis übernommen werden (siehe dazu auch Abbildung 8.7). Diese wird zentral von der Kernkomponente des Pixel-Advisors verwaltet und muss nach einer Änderung von der

[9]Ein roter Balken bedeutet, dass mindestens einer der Kinderknoten ein Problem hat, während ein gelbes Warndreieck vorhandene Aktivierungen auf dem Knoten selbst symbolisiert.

[10]Im Allgemeinen müssen mehrere Fakten einen gewissen Wert annehmen, damit eine Regel aktiviert wird.

Rule-Engine neu eingelesen werden, was den Betrieb des Expertensystems kurzzeitig unterbricht. Daher nimmt der Server der Kernkomponente zunächst nur die von jedem Client vorgenommenen Änderungen an den Salience-Werten der vorhandenen Regeln entgegen und speichert diese Änderungen in einem Zwischenspeicher.

Ein autorisierter Expertensystem-Administrator kann dann aus dem Userinterface-Client heraus eine Übernahme der im Zwischenspeicher angesammelten Nutzer-Rückmeldungen in die Regelbasis veranlassen. Erst danach treten die damit zusammenhängenden Wahrscheinlichkeitsänderungen in Kraft.

8.4.2 Der Regeleditor

Neben der für den Nutzer des Pixel-Advisors sicherlich wichtigsten Aktivierungsansicht, bietet das Userinterface auch eine Regelansicht, in der bestehende Regeln modifiziert und neue Regeln angelegt werden können.

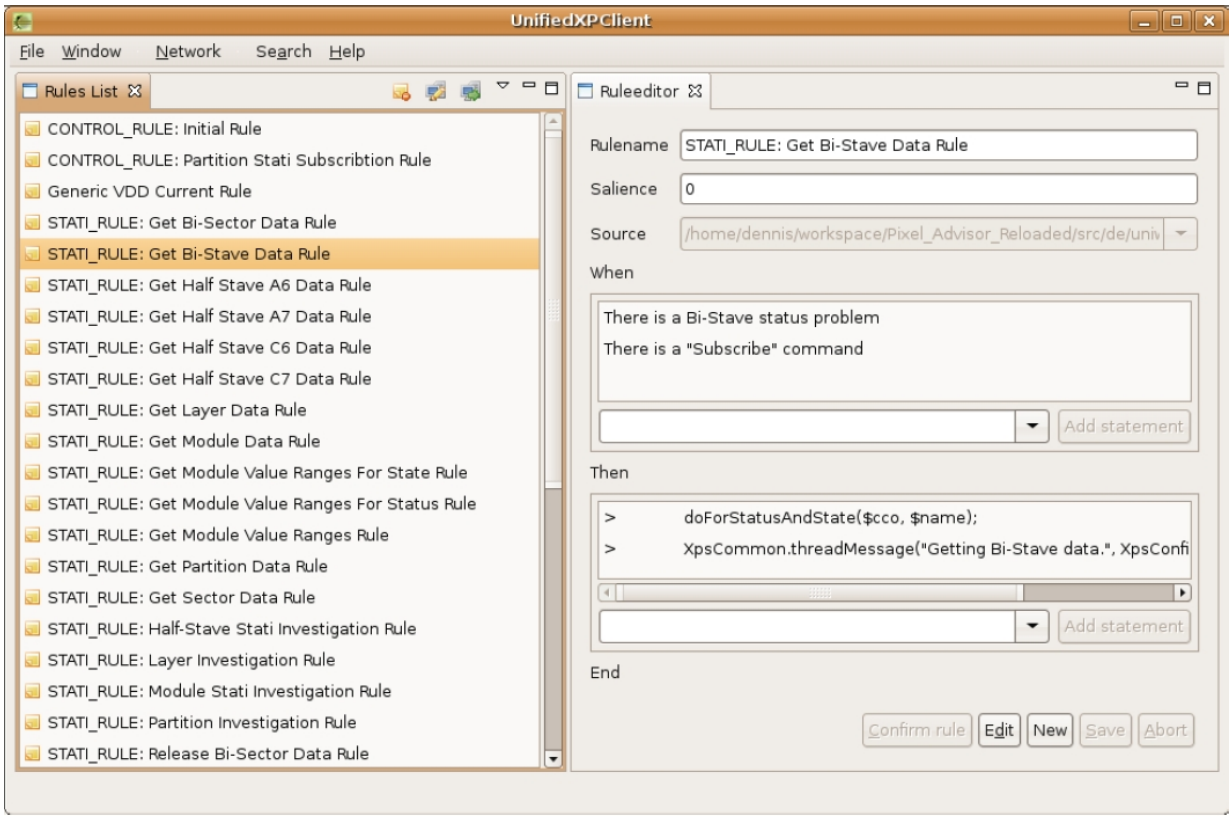


Abbildung 8.9: *Regelansicht des UnifiedXP-Client. Diesmal wurde das Bildschirmfoto auf einem Rechner mit Ubuntu-Linux-Betriebssystem gemacht.*

Abbildung 8.9 zeigt die Regelansicht des Pixel-Advisor, bzw. des UnifiedXP-Client. Auf der linken Seite befindet sich dabei eine “Rules List”, die die Regeln der Regelbasis

der Kernkomponente anzeigt. Dabei kann über Filter gewählt werden, ob alle Regeln oder lediglich bestimmte Regelsätze, wie beispielsweise die Kontroll-Regeln angezeigt werden sollen.

Wählt der Nutzer eine Regel in der Rules List aus, so wird diese im "Rule Editor" angezeigt, der sich auf der rechten Seite des Regelansicht-Fensters befindet. In diesem Editor wird die Regel in ihre einzelnen Blöcke, die Prämissen und Konklusionen, aufgespalten, die dann einzeln editiert werden können. Dazu kann der Nutzer entweder beliebigen Text selbst eingeben oder über die in Abbildung 8.9 sichtbaren Auswahlboxen bereits bestehende DSL-Snippets auswählen.

Sobald eine bestehende Regel editiert oder eine neue angelegt wurde, wird diese Änderung an der Regelbasis durch ein Modifikations-Icon in der Rules List verdeutlicht. Analog zum Vorgehen bei der Berücksichtigung der vom Nutzer modifizierten Regel-Salience, wird auch im Falle der Modifikation einer Regel diese nicht direkt in die Regelbasis übernommen, sondern zunächst auf Serverseite zwischengespeichert.

Ein Administrator trägt dann die Verantwortung, die eingegangenen Modifikations-"Vorschläge" auf ihre logische und syntaktische Korrektheit zu prüfen, eventuell zu berichtigen und schließlich in die Regelbasis zu übernehmen, was seinerseits ein erneutes Einlesen der Regelbasis durch die Rule-Engine erforderlich macht.

Obwohl dieses Vorgehen die Administratoren des Expertensystems dazu zwingt, in regelmäßigen Abständen die vorgeschlagene Regelbasis zu überprüfen, bringt es doch zwei wesentliche Vorteile mit sich:

- **Robustheit der Regeln:** *Da JEDE Regel vom Administrator VOR Übernahme in die tatsächliche Regelbasis abgezeichnet werden muss, ist sichergestellt, dass nur solche Regeln übernommen werden, deren Syntax und Logik überprüft wurden.*

- **Geringe Nutzerhemmschwelle:** *Da der vom Nutzer als Regelvorschlag eingegebene Text NICHT direkt in die Regelbasis übernommen wird, muss dieser NICHT der eigentlichen Regelsyntax, bzw. den zur Verfügung gestellten Schnipseln natürlicher Sprache entsprechen. Damit wird ermöglicht, dass ein Nutzer sein Expertenwissen in normaler Sprache eingibt, das dann vom Administrator in eine syntaxkonforme Regel umgewandelt werden kann. Obwohl dies zunächst zusätzliche Arbeit vom Administrator verlangt, reduziert es für den Nutzer die Hemmschwelle, das eigene Wissen in den Pixel-Advisor einzugeben.*

Da die Akquisition einer ausreichenden Wissensbasis zu den wichtigsten Aufgaben und größten Herausforderungen eines Expertensystems gehört, wird eine höhere Belastung der Administratoren zu Gunsten einer möglichst weitreichenden Entlastung der Detektorexperten und Nutzer in Kauf genommen.

8.5 Die Kernkomponenten

Wie bereits in den vorangegangenen Abschnitten erwähnt, basiert der Pixel-Advisor auf einer Client-/Server-Architektur. Dabei verbinden sich beliebig viele Userinterface-Clients mit den als Server fungierenden Kernkomponenten des Pixel-Advisor. Diese bestehen im Wesentlichen aus den Schnittstellen zur Ankopplung der Datennahmekomponente, der Regelbasis und der Rule-Engine, wobei letztere gemäß der in Abschnitt 8.1 eingeführten Einteilung, Teil der Kernkomponenten ist.

Um Daten mit den Datennahmekomponenten des GridXP bzw. des Pixel-Advisor austauschen zu können, wurde ein gemeinsames Datenformat in Form einer Containerklasse entwickelt. Die ursprünglich aus DIM und RGMA[11] gewonnenen Fakten werden dazu in Instanzen dieser Containerklasse gespeichert und dann an den UnifiedXP-Kern übergeben, der alle an ihn übermittelten Fakten verwaltet. Von dort aus werden die Fakten dann in die Rule-Engine übermittelt, wobei diese auf Grund der Möglichkeit des direkten Java-Bean[12]-Zugriffs lediglich eine Referenz auf das jeweilige Faktum innerhalb des UnifiedXP-Kerns erhält.

Über den Java-Mechanismus der Listener[13] ist es weiterhin nicht notwendig, dass Kern oder Rule-Engine über Modifikationen an Fakten informiert werden, da dies bei Faktenänderung durch die Datennahmekomponente automatisch geschieht.

Um den Sicherheitsanforderungen des Grid-Einsatzes zu entsprechen, wird die Kommunikation zwischen dem Kern-Server und den Userinterface-Clients über den SSL[14]-Mechanismus verschlüsselt und gegen unbefugte Manipulation gesichert.

Da die von JBoss Rules aus den Regeldateien eingelesenen Regelobjekte nicht über die notwendige Eigenschaften für den Einsatz im Regeleditor des Userinterfaces verfügen, werden die Regeldateien vom UnifiedXP-Kern ebenfalls geparst und in einzelne Java-Regelobjekte zerlegt.

Zwischen Client und Server werden dann, neben Steuerbefehlen, ausschließlich die Fakten-, Aktivierungs- und Regelobjekte ausgetauscht. In Zusammenhang mit den in Abschnitt 8.4 getroffenen Vorkehrungen führt dies dazu, dass weder Userinterface noch

[11] **R**elational **G**rid **M**onitoring **A**rchitecture

[12] Bei Java-Beans handelt es sich um Java-Klassen, die unter anderem über standardisierte Konstruktoren sowie Zugriffsmethoden auf die Membervariablen verfügen. Weitere Definitionseigenschaften werden von den hier genutzten Beans zwar erfüllt (z.B. Serialisierbarkeit) sind aber für die hier angeführten Mechanismen nicht von Bedeutung.

[13] Listener sind wichtiger Bestandteil der "ereignisgetriebenen" Programmierung. Dabei wird der Programmablauf nicht durch den üblichen Verarbeitungsstapel, sondern durch die Reaktion auf das Eintreten bestimmter Ereignisse gesteuert. Listener werden dabei über bestimmte Ereignisse informiert, sofern sie sich vorher als Interessent des Ereignisses angemeldet haben. Ein einfaches Beispiel für diesen Mechanismus ist das Drücken eines GUI-Knopfes, woraufhin dieser seine Listener informiert, dass er gedrückt wurde.

[14] **S**ecure **S**ockets **L**ayer

Kernkomponenten des Pixel-Advisor oder GridXP spezifisches Wissen über das Anwendungsgebiet benötigen und daher im Rahmen der UnifiedXP-Komponenten gemeinsam genutzt werden.

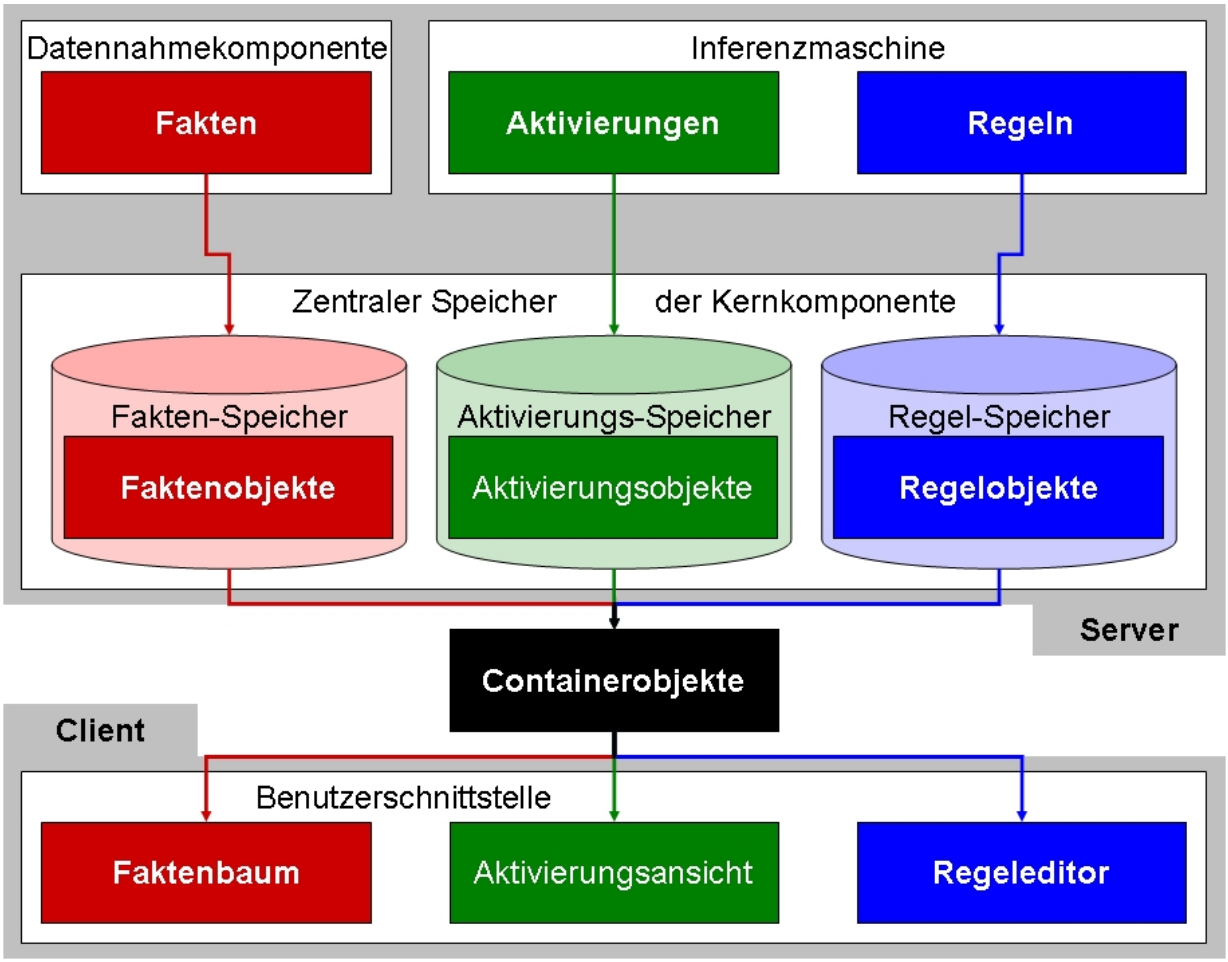


Abbildung 8.10: *Schematischer Darstellung der Nutzung eines gemeinsamen Containerformats für die Kommunikation zwischen Server und Client.*

Abbildung 8.10 zeigt eine schematische Darstellung des Datenaustauschs zwischen Server und Client mit Hilfe der erwähnten Containerobjekte. Diese fungieren als Abstraktionsschicht für die Fakten-, Aktivierungs- und Regelobjekte, was die Nutzung eines einzigen Datenaustauschformats zwischen Client und Server ermöglicht. Im Gegensatz zu Fakten und Aktivierungen können modifizierte Regelobjekte dabei durch den Regeleditor vom Userinterface zurück an die Kernkomponente übertragen werden, wo sie zunächst im bereits in Abschnitt 8.4 erwähnten Zwischenspeicher der Kernkomponente gespeichert werden. Von dort aus können die Regel dann durch einen berechtigten Nutzer an die Regelbasis und damit die Inferenzmaschine übermittelt werden.

Während Erzeugung und Modifikation der Regelobjekte durch den Anwender ausgelöst werden, in dem dieser die Regelbasis lädt oder modifiziert, werden Änderungen an den Fakten- und Aktivierungsspeichern im laufenden Betrieb durch Werteänderungen der Betriebsparameter des Pixeldetektors ausgelöst. Damit die Benutzerschnittstelle über solche Änderungen informiert wird, muss der Server entsprechende Vorkehrungen treffen. Da die ursprüngliche Implementation dieses Mechanismus sich erst im Betrieb als unzureichend erwies und daraufhin geändert werden musste, soll er in Abschnitt 9.1.2 beschrieben werden.

Kapitel 9

Einsatz des Pixel-Advisors am CERN

Nachdem die vorangehenden Kapitel das Einsatzgebiet und die Implementation des Pixel-Advisors behandelt haben, summiert dieses Kapitel die Erfahrungen, die bisher durch den Betrieb der Software am CERN gewonnen wurden.

Der Pixel-Advisor wird dabei seit September 2008 von den Pixel-DCS-Experten eingesetzt, um in Zusammenarbeit mit der Pixel-DCS-FSM den Detektor während des laufenden Betriebs zu überwachen und Ursachen für mögliche Fehler zu finden. Da dies der erste Einsatz der Software unter realen Bedingungen war, konnten wertvolle Erkenntnisse bezüglich vorhandener Schwachpunkte und Stärken gesammelt werden.

9.1 Verbesserungen

Einige der in den vorangehenden Kapiteln angeführten Designentscheidungen führten, unter den im Vergleich zu den Testsystemen geänderten Bedingungen, zu mehr oder weniger schweren Problemen mit der Software. Diese Probleme und die im einzelnen umgesetzten notwendigen Änderungen werden in den folgenden Abschnitten erläutert.

9.1.1 Erklärung des FSM-Zustands

Nach dem offiziellen Beginn des ATLAS-Schichtbetriebs zeigte sich, dass die überwiegend aus Nicht-DCS-Experten bestehenden DCS-Shifter Probleme bei der Interpretation der von der FSM bereitgestellten Daten hatten. Insbesondere die Frage, warum ein gegebenes Modul oder eine Ausleseeinheit in einer bestimmten State/Status-Kombination war, konnte häufig nur durch Rücksprache mit dem DCS-Experten der Bereitschaft hatte geklärt werden.

Aus Sicht der DCS-Experten war es dabei besonders ärgerlich, dass diese Information prinzipiell in Form einer Beschreibung innerhalb der Datenpunkte der FSM vorlag, jedoch bisher nicht im Userinterface angezeigt werden konnte.

Daraus ergab sich eine erste Möglichkeit für das Expertensystem, dem Schichthabenden eine wichtige Hilfestellung zu bieten, indem es die State/Status-Kombination der einzelnen Detektorkomponenten erklärt.

Der dazu notwendige Ablauf basiert auf der Tatsache, dass jede FSM-Device-Unit ihren State und Status aus dem Vergleich der vorliegenden Betriebsparameter mit einem Satz von Grenzwerten bezieht. Für jede Kombination aus State und Status existiert dabei mindestens ein Satz dieser Grenzwerte für jeden Betriebsparameter, sowie eine zugehörige Beschreibung. Abbildung 9.1 zeigt einen solchen Satz von FSM-Bestimmungsparametern für ein beliebiges Detektormodul.

-0.1	0.1	-0.1	0.05	-1	1	-0.1	4	OFF	OFF	OK	
-0.1	0.1	-0.1	0.05	-1	1	-0.1	4	DISABLED	OFF	OK	
0.35	0.38	0.18	0.2	10	700	-0.1	4	ON	READY	OK	FEbyFE
0.65	1	1	1.45	10	700	-0.1	4	ON	READY	OK	
1	1.8	1	1.45	10	700	-0.1	4	ON	READY	WARNING	NOISY
0.28	0.5	0.05	0.16	10	700	-0.1	4	ON	ON	OK	
0.42	0.65	-0.1	1.45	10	700	-0.1	4	ON	ON	WARNING	LOST_CLOCK
0.5	0.65	0.05	1.45	-1	10	-0.1	4	OFF	QUIET	OK	?
0.29	1.8	0.05	1.45	-1	10	-0.1	4	OFF	QUIET	OK	DAQ_ACTION_REQUIRED
0.5	0.65	0.05	1.45	-1	10	-0.1	4	DISABLED	QUIET	OK	?
0.1	1.65	0.05	1.45	-1	10	-0.1	4	DISABLED	QUIET	OK	DAQ_ACTION_REQUIRED
-0.1	0.1	-0.1	0.05	10	700	-0.1	4	ON	UNDEFINED	WARNING	ONLY_HV_ON
0.1	0.3	-0.1	1.45	-1	700	-0.1	4	ON	UNDEFINED	WARNING	NO_CLOCK
0.3	1	0.05	1.45	-1	700	-0.1	4	ON	UNDEFINED	OK	RAMPING
1.8	2	0.1	1.6	-1	700	-0.1	4	ON	UNDEFINED	ERROR	OVERCURRENT
1.8	2	-0.1	1.6	-1	700	-0.1	4	ON	UNDEFINED	ERROR	OVERCURRENT_Vdd
-0.1	2	1.45	1.6	-1	700	-0.1	4	ON	UNDEFINED	ERROR	OVERCURRENT_Vdda
-0.1	1.8	-0.1	1.45	-1	700	-0.1	4		UNDEFINED	WARNING	
-0.1	2	-0.1	1.6	-1	700	-0.1	4		UNDEFINED	WARNING	
-0.1	2.2	-0.1	1.8	-1	700	-0.1	4	UNKNOWN	UNKNOWN	ERROR	HV_UNKNOWN
-0.1	2.2	-0.1	1.8	-1	700	-0.1	4		UNDEFINED	ERROR	
-0.1	2.4	-0.1	2	-1	700	-0.1	4		UNDEFINED	FATAL	

Abbildung 9.1: *FSM-Bestimmungsparameter eines beliebigen Detektormoduls. Aus dem Vergleich der tatsächlich vorliegenden Betriebsparameter mit den gegebenen Schranken bestimmt die FSM die aktuelle State/Status-Kombination.*

Das von der Pixel-FSM genutzt Format setzt sich dabei pro Zeile aus den folgenden Einträgen zusammen:

$$|IDD_{min}|IDD_{max}|IDDA_{min}|IDDA_{max}|HV_{min}|HV_{max}|IHV_{min}|IHV_{min}|HVState|State|Status|Description|$$

Sofern die vier Modulparameter Digitalstrom IDD, Analogstrom $IDDA$, Hochspannung HV und Sperrstrom IHV in die durch min und max angegeben Grenzen fallen, nimmt das entsprechende Detektormodul den durch $State : Exp$ und $Status$ bezeichneten FSM-Zustand an. Da diese FSM-Bestimmungsparameter für jede Device-Unit in speziellen PVSS-Datenpunkten abgespeichert sind, nutzt der Pixel-Advisor seine durch DIM gegebene, direkte Zugriffsmöglichkeit auf diese Datenpunkte, um die resultierende Beschreibung in Erfahrung zu bringen. Im Falle einer problematischen Device-Unit wird

dementsprechend eine Aktivierung erzeugt und im GUI des Pixel-Advisors wird die FSM-Beschreibung der eigenen State/Status-Kombination eingeblendet.

Neben dem dadurch verbesserten Verständnis des Schichtpersonals für den Zustand der FSM ist ein weiterer wichtiger Punkt dieses Vorgehens, dass somit automatisch die aktuell genutzte Konfiguration des Pixeldetektors berücksichtigt werden kann. Da prinzipiell jedes der 1744 Detektormodule und jede der 272 Ausleseeinheiten leicht unterschiedliche Betriebsparameter und damit auch FSM-Grenzwerte haben kann, die sich darüberhinaus auch noch von Konfiguration zu Konfiguration ändern können, wäre die Erklärung des FSM-Zustandes ohne Zuhilfenahme der angesprochenen Datenpunkte schwer möglich.

Während die Anbindung des Expertensystems an die Verkabelungs- und Konfigurationsdatenbanken anfänglich als notwendig aber vorerst kaum realisierbar eingeschätzt wurde, zeigte sich damit, dass der direkte Zugriff auf die von DCS momentan genutzten Parameter in diesem Fall die bessere Lösung war. Dieser gewährleistet, dass die vom Expertensystem genutzten Werte mit denen des DCS übereinstimmen; die Gefahr eine falsche Version der Datenbankeinträge ("Tags") zu nutzen besteht damit nicht.

9.1.2 Feuern der Werteregeln

Das vorherige Beispiel der Erklärung des FSM-Zustands durch das Expertensystem zeigt, dass ein regelbasiertes System nicht zwangsläufig aus sehr vielen Regeln bestehen muss, um positive Ergebnissse zu erzielen. Gerade dabei zeigte sich aber auch eine der wesentlichen Schwächen des ursprünglichen Konzepts des Pixel-Advisors.

Wie bereits in Abschnitt 8.4.1 erwähnt, ist eines der Hauptziele des Expertensystems, die auftauchenden Probleme und daraus resultierenden Lösungsvorschläge der Detektorgeographie folgend, übersichtlich darzustellen. Sobald ein Teil des Detektors in einen problematischen FSM-Zustand wechselt, werden die zugehörigen Lösungsvorschläge und Erklärungen auf dem entsprechenden Knoten in dem Faktenbaum dargestellt (siehe auch Abbildung 8.8).

Um dies zu erreichen, benötigt man zwei Informationen, nämlich die Benachrichtigung über das erstmalige Erfüllen des Bedingungsteils einer Regel (also das Erzeugen einer Aktivierung), sowie über das erstmalige Verletzen dieses Bedingungsteils (das Löschen der Aktivierung). Der ursprüngliche Mechanismus nutzte dabei Java-Listener, um sich mit der Agenda der Regelmaschine zu verbinden und auf entsprechende Benachrichtigungen zu reagieren (siehe Abbildung 9.2). Problematisch bei diesem Ansatz ist die Tatsache, dass eine Aktivierung innerhalb der Regelmaschine auch dann als gelöscht gilt, wenn die aktivierte Regel von der Regelmaschine gefeuert wurde.

In einem einfachen Beispiel, einer zu hohen Temperatur führt dies dazu, dass zum Zeitpunkt t_0 eine Aktivierung auf Grund des Überschreitens einer zulässigen Maximaltemperatur erzeugt wird. Wird die aktivierte Regel dann von der Rule-Engine gefeuert,

also ihr Anweisungsteil ausgeführt, so wird die Aktivierung unmittelbar gelöscht. 5 Sekunden später, zum Zeitpunkt t_1, überträgt das DCS den nächsten Temperaturwert an das System, der, sollte die zu hohe Temperatur nach wie vor vorliegen, eine erneute Aktivierung erzeugt. Der beschriebene Vorgang würde sich also alle paar Sekunden wiederholen, Aktivierungen würden in den Faktenbaum eingetragen, andere gelöscht.

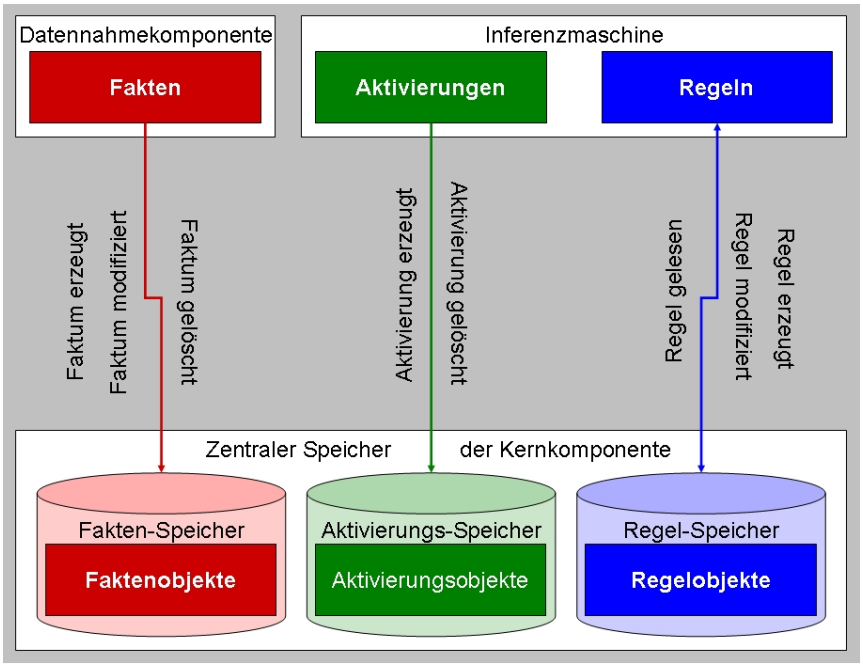


Abbildung 9.2: *Ursprüngliche Implementation des Benachrichtigungsmechanismus. Damit das Erzeugen und Vernichten von Aktivierungsobjekten wie gewünscht funktioniert, dürfen die zugehörigen Regeln nicht gefeuert werden.*

Da dies nicht das gewünschte Verhalten darstellt, nachdem eine Aktivierung nur dann aus dem Baum entfernt werden soll, wenn die Ursache auch wirklich behoben ist, sah der zunächst implementierte Ansatz vor, die Werteregeln **nicht** zu feuern. Damit können dann aber auch die in den Regeln enthaltenen Konklusionen nicht ausgeführt werden. Um trotzdem Erklärungen und Lösungsvorschläge innerhalb einer Regel formulieren zu können, wurden diese durch die einfache Zeichenkette *Solution* : "Lösungsvorschlag" gekennzeichnet. Das Userinterface konnte damit die unter "Lösungsvorschlag" eingetragene Lösung zu jeder Regel durch Parsen der Regeldatei auslesen und bei Aktivierung der Regel anzeigen.

Dieser Trick erschien zunächst vielversprechend und ermöglichte das gewünschte Verhalten, da die Regeln nun nicht mehr gefeuert werden mussten, um eine Lösung anzuzeigen und damit die erzeugten Aktivierungen so lange bestehen blieben, bis der Bedingungsteil der Regel durch die vorhandenen Fakten nicht mehr erfüllt war. Der Nachteil dieses Vorgehens ist jedoch, dass es nicht möglich ist, im Lösungsvorschlag auf die aktuellen Variablenwerte, die zur Aktivierung der Regel geführt haben, einzugehen. Diese sind zum Zeitpunkt des Einlesens durch den Parser ja noch unbekannt. Man kann im erwähnten

Beispiel also lediglich berichten, dass die Modultemperatur die eingestellte Alarmschwelle von beispielsweise 27.0 °C überschritten hat, nicht aber welchen Wert die Temperatur aktuell tatsächlich hat.

Auch dies war vorerst unproblematisch, da die aktuelle Temperatur, sowie alle anderen Fakten, die zu einer Aktivierung führen, in der Aktivierungsansicht zusätzlich dargestellt werden (siehe Abbildung 8.8). Die Unzulänglichkeit, diese nicht im selben Textfeld innerhalb des Lösungsvorschlags darstellen zu können wurde daher während des ursprünglichen Designs als nebensächlich betrachtet.

Im Zusammenhang mit den im letzten Abschnitt beschriebenen Erklärungsregeln für den FSM-Zustand zeigte sich aber, dass es dadurch notwendig wurde, für jede anzuzeigende Erklärung wie etwa "Module is in UNDEFINED/WARNING because it is NOISY" oder "Module is in UNDEFINED/ERROR because of OVERCURRENT_vdd" eine eigene Regel anzulegen.

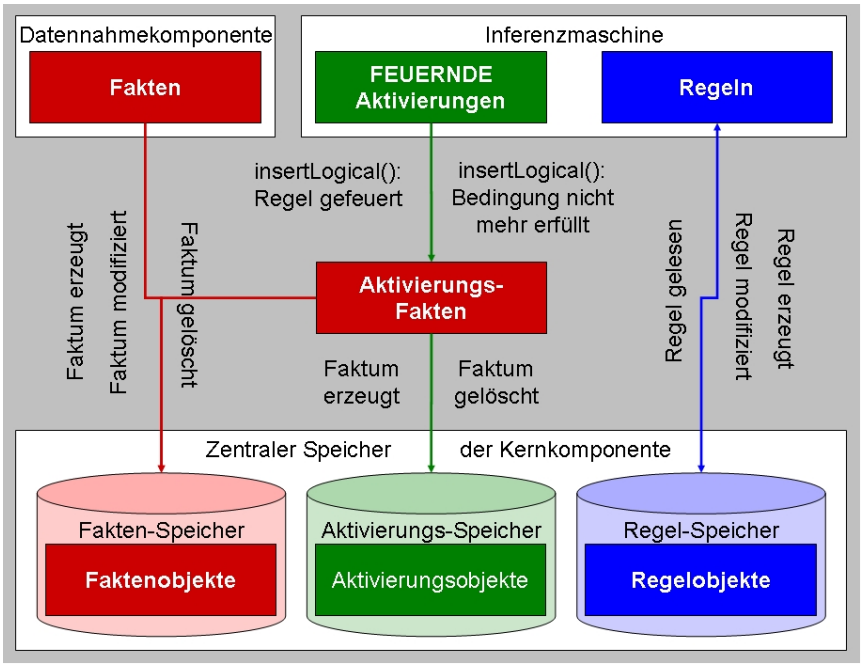


Abbildung 9.3: *Der überarbeitete Benachrichtigungsmechanismus. Er erlaubt das Feuern der Regeln, indem Aktivierungsobjekte über insertLogical() mit den Regeln verbunden werden. Die eigentliche JBoss Rules Aktivierung der Regel wird nicht mehr genutzt.*

Um diese und damit zusammenhängende Beschränkungen aufzuheben, wurde daher der Benachrichtigungsmechanismus dahingehend geändert, dass diese Regeln gefeuert werden können und damit den Zugriff auf ihre Variablen erlauben. Dazu wurde der Mechanismus "logischer Fakten" innerhalb der JBoss Rules Regelmaschine genutzt, der es erlaubt ein logisches Faktum über den Befehl "insertLogical()" in die Wissensbasis einzufügen, sobald eine Regel gefeuert wird. Dieses Faktum bleibt dann solange bestehen, bis die Bedingung

der Regel verletzt ist und wird dann automatisch entfernt. Dementsprechend wurden die bestehenden Regeln modifiziert und die erwähnten Java-Listener wurden so abgeändert, dass sie nicht mehr auf das Erzeugen bzw. Löschen von Aktivierungen, sondern das Anlegen und Löschen von logischen Fakten reagieren (siehe Abbildung 9.3). Abbildung 9.4 zeigt die Bedingungen für das Erzeugen und Vernichten eines Pixel-Advisor-Lösungsvorschlags mit Hilfe von Aktivierungen einerseits und logischen Fakten andererseits. Mit dem neuimplementierten Verfahren ist es nun möglich, aktuelle Werte der Regelvariablen in den Lösungsvorschlag aufzunehmen, was dazu führt, dass die erwähnten FSM-Zustandsregeln durch eine einzige Regel ersetzt werden konnten, die die Zustandsbeschreibung in den Lösungsvorschlag integriert.

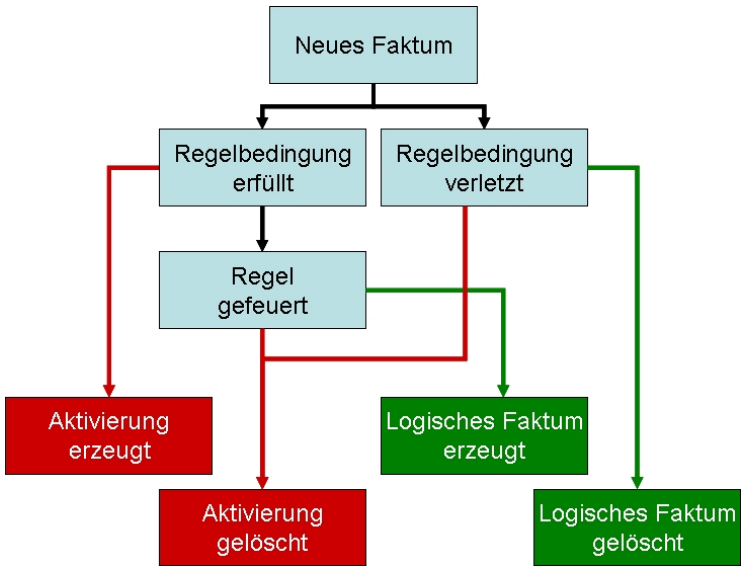


Abbildung 9.4: *Vergleich der Bedingungen für das Erzeugen/Vernichten von JBoss Rules Aktivierungen bzw. logischen Fakten.*

9.1.3 Tiefensuche

Eines der Probleme, das im Testbetrieb mit dem FSM-Simulator als auch bei ersten Tests am CERN nicht aufgefallen war, hängt mit der selbstauferlegten DIM-Maximallast des Pixel-Advisors zusammen. Wie bereits in Abschnitt 8.2.1 beschrieben, ist es möglich, dass die maximale Anzahl von DIM-Verbindungen erreicht ist, bevor die Datennahmekomponente zur Ursache des Problems vordringen konnte. Während diese Situation im Simulator und am CERN-SR1-Testsystem auf Grund der fehlenden Größe der Systeme nicht weiter problematisch war, stellte sich heraus, dass sie für den Pixeldetektor selbst zur Regel wurde, weil bereits etwa 10 problematische und ungünstig über den Detektor verteilte Module ausreichen, um die DIM-Begrenzung zu überschreiten. Abbildung 9.5 zeigt die Situation schematisch anhand des bereits bekannten FSM-Baumes.

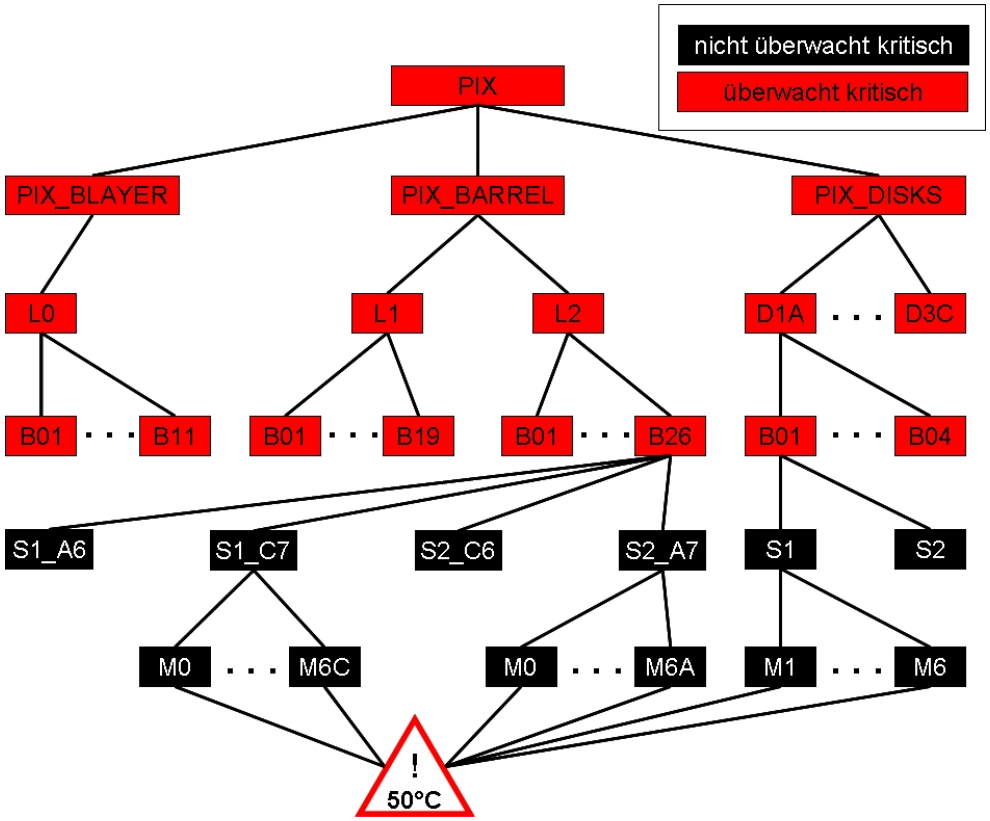


Abbildung 9.5: *Die bisher ausschließlich genutzte Breitensuche. Befindet sich eine genügend große Zahl an FSM-Control-Units in einem kritischen Überwachungszustand, so verhindert die DIM-Begrenzung das Vordringen zu den problematischen Device-Units. Die Situation entspannt sich erst wieder, wenn Teile des Detektors ohne Hilfe des Expertensystems in einen unkritischen Zustand überführt werden.*

Da die Erhöhung dieser Maximalzahl an DIM-Items das Problem nur verschiebt und eine zu hohe Maximalzahl außerdem gegen die in Abschnitt 6.1.2 beschriebenen Anforderungen verstößt, wurde eine Alternative implementiert. Dabei handelt es sich um einen zweiten Algorithmus zur Steuerung des Datenflusses, also der Reihenfolge in der die verschiedenen Betriebsparameter durch den Pixel-Advisor vom Detektorkontrollsystem angefordert werden. Im Gegensatz zur bisherigen “Breitensuche”, die nur dann zum Ziel führt, wenn die Anzahl der zu überwachenden DIM-Items klein genug ist, durchsucht diese “Tiefensuche” den FSM-Baum Zweig für Zweig. Selbst in der problematischen, weil über die meisten FSM-Knoten verfügenden “L2” Zylinderlage benötigt die Tiefensuche höchstens 50 DIM-Items, um bis zum ersten problematischen Modul vorzustoßen und zwar unabhängig davon, wieviele Probleme in den restlichen Ästen des FSM-Baumes vorliegen (siehe Abbildung 9.6).

Ein weiterer Vorteil der Tiefensuche ist die damit verbundene Möglichkeit, eine Problemsuche gezielt durch den Nutzer des Expertensystems anstoßen zu lassen. Obwohl zum

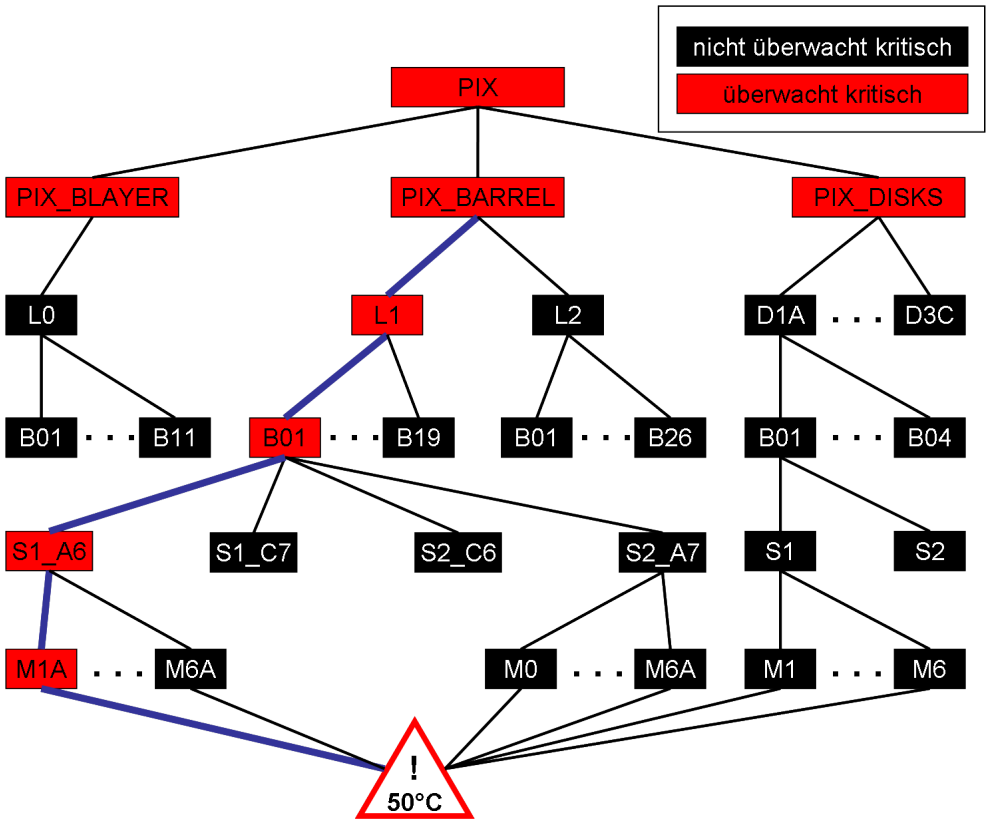
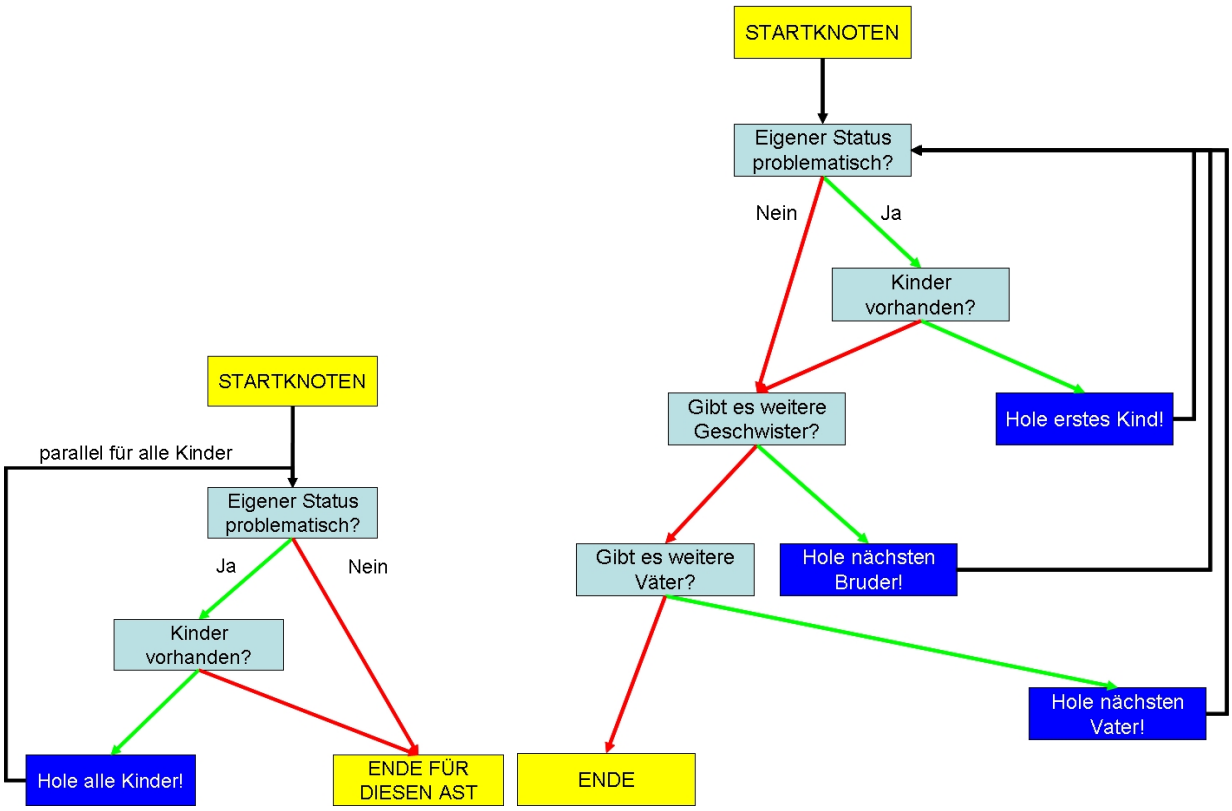


Abbildung 9.6: *Beispiel zur Tiefensuche. Ist ein problematischer FSM-Knoten entdeckt, werden zunächst die Daten seines ersten Kindes angefordert, bevor mit seinen Geschwisterknoten fortgefahren wird. Dadurch ist sichergestellt, dass zumindest die ursächlichen Device-Units einiger Problemknoten untersucht werden können, selbst wenn die DIM-Begrenzung die Datennahme aller problematischen Knoten unterbindet.*

Zeitpunkt dieser Arbeit noch nicht implementiert, könnte dieser dazu einzelne Knoten des Faktenbaumes auswählen und das System dazu veranlassen, deren Daten zu holen.

Abbildung 9.7 zeigt die vereinfachten Flussdiagramme der beiden Suchalgorithmen. Während die einfache Breitensuche lediglich 3 Funktionsaufrufe für die Datennahme aller direkten Kinder eines Knotens benötigt, sind im Falle der Tiefensuche bereits bis zu 5 Funktionsaufrufe pro direktem Kind notwendig. Ein weiterer wesentlicher Punkt, der dazu beiträgt, dass der Tiefensuch-Algorithmus deutlich langsamer arbeitet, als die Breitensuche ist darin begründet, dass der DIM-Mechanismus die Bündelung von Anfragen in einem einzigen Befehl an den Expertensystem-Kommandodatenpunkt erlaubt. Diese Bündelung wird von der Breitensuche an der Stelle ausgenutzt, an der die Daten “parallel für alle Kinder” geholt werden, was intern nur ein einziges DIM-Kommando benötigt. Die Tiefensuche hingegen kann diesen Mechanismus nicht nutzten, da erst nach Abfragen des aktuellen Knotens entschieden werden kann, welches der nächste notwendige Knoten ist.



(a) Der Algorithmus der Breitensuche ist sehr einfach und nutzt nur wenige Funktionsaufrufe (blau). Der entscheidende Geschwindigkeitsunterschied ergibt sich aus der Nutzung gebündelter DIM-Anfrage (dunkelblau) für alle Kinder eines Knotens.

(b) Die Tiefensuche ist komplizierter und benötigt vor allem wesentlich mehr Funktionsaufrufe pro Durchlauf. Die DIM-Aufrufe (dunkelblau) beziehen sich hier jeweils nur auf genau einen Knoten.

Abbildung 9.7: *Vergleich der beiden Suchalgorithmen.*

Auf Grund der vorhandenen Vor- und Nachteile beider Suchstrategien kann der Nutzer diese selbst auswählen.

9.1.4 Server-Shutdown

Obwohl die bereits mehrfach erwähnte DIM-Item-Maximalgrenze sicherstellt, dass der Pixel-Advisor zu keinem Zeitpunkt mehr als die angegebene Anzahl an DIM-Items nutzt, kam es während der ersten Betriebstage zu Situationen, in denen die Anzahl der von DIM veröffentlichten und durch das Präfix “xpt_” gekennzeichneten, ausschließlich für das Expertensystem veröffentlichten DIM-Items bei der vierfachen Anzahl lag. Dadurch alarmiert gelang es, den Fehler auf ein Zusammenspiel zweier Probleme zurückzuführen.

Einerseits funktionierte der eingebaute Abschaltmechanismus des Expertensystem-Servers offensichtlich nicht. Dieser soll dafür sorgen, dass beim Herunterfahren des Servers die Veröffentlichungen aller verbliebenen und mit "xpt_" beginnenden DIM-Items zurückgenommen werden. Ein abruptes Beenden des Servers über einen "kill"-Befehl scheint diesen Mechanismus zu umgehen, was wahrscheinlich nur durch einen, von PVSS überwachten, "alive"-Mechanismus abzufangen wäre. Das zweite ursächliche Problem bestand darin, dass der Server selten von den Clients ordnungsgemäß heruntergefahren wurde. Dazu sollte ursprünglich auf Client-Seite ein spezielles "Shutdown Server"-Kommando gegeben werden, was auf Grund menschlicher Bequemlichkeit selten geschah.

Die implementierte Verbesserung besteht darin, dass der Server nun automatisch den Befehl, alle DIM-Items freizugeben an das DCS sendet, sobald sich der letzte mit ihm verbundene Client abmeldet, ohne dass dieser dazu ein spezielles Kommando senden müsste. Im Umkehrschluss führt dies dazu, dass der Server bei der Verbindung des ersten Clients zunächst wieder alle benötigten DIM-Items anfordern muss, so dass der erste Verbindungsaufbau länger dauert als in der ursprünglichen Version.

Für die Zukunft wäre sicherlich eine zusätzliche Sicherheitsebene in Form eines den Serverzustand überwachenden PVSS-Skripts wünschenswert, das im Falle eines abrupten Serverabsturzes die verbliebenen DIM-Items freigibt. Noch sinnvoller wäre aber in Anbetracht der zentralen Bedeutung des DIM-Protokolls für die LHC-Experimente sicherlich eine DIM-seitige Sicherung vor zu hoher Last, beispielsweise durch das Ablehnen zusätzlicher Verbindungen.

9.1.5 Speichern der Regelvorschläge

Eine sicherlich optimistische, ursprüngliche Annahme war, dass die Serveranwendung von niemandem beendet werden würde, bevor ein mit Administratorrechten versehener Nutzer die zuvor im Regel- und Bewertungszwischenspeicher abgelegten Änderungen in die Regelbasis übernehmen konnte (siehe Abbildung 8.7). Wie der vorherige Abschnitt zeigt, konnte nicht länger davon ausgegangen werden, dass die Vorkehrung, nur die Administrator-Clients mit einem "Shutdown-Server"-Knopf zu versehen, einen ausreichenden Schutz vor einem vorzeitigen Server-Stop bot.

Da die von den Nutzern eingegebenen Änderungen aber keinesfalls ungelesen gelöscht werden dürfen, wurde der in Abbildung 8.7 gezeigte Ablauf dahingehend verändert, dass die von einem Client übermittelten Regel- und Bewertungsänderungen nun direkt in entsprechende Dateien geschrieben werden, mit denen der Server-Cache synchronisiert wird (siehe Abbildung 9.8). Dadurch ist gewährleistet, dass auch ein unbeabsichtigtes, abruptes Stoppen des Servers die eingegebenen Vorschläge nicht löscht.

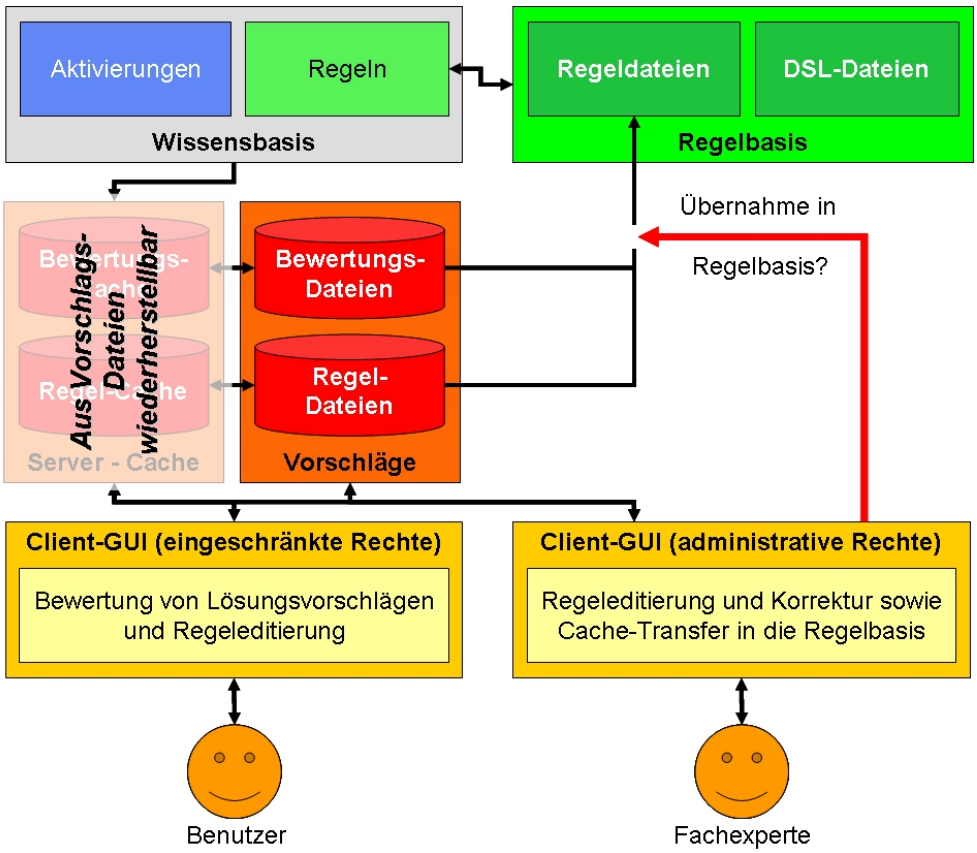


Abbildung 9.8: *Funktionsprinzip des Server-Caches unter Hinzunahme der Vorschlagsdateien. Solange der Server läuft werden die in den Dateien enthaltenen Bewertungs- und Regelvorschläge mit dem eigentlichen Server-Cache synchronisiert. Im Falle eines Serverabsturzes kann der Cache aus den Vorschlagsdateien wiederhergestellt werden.*

9.1.6 Simulator

Im Gegensatz zu den leicht manipulierbaren Rahmenbedingungen der Testumbgebungen, war es sehr schwierig neue Regeln im realen System zu überprüfen. Hatte man beispielsweise gerade eine neue Regel bezüglich kritischer Optoboard-Temperaturen entworfen, so gab es keine andere Möglichkeit diese prüfen, als darauf zu warten, dass das entsprechende Problem tatsächlich in Form einer kritischen Optoboard-Temperatur auftrat.

Um diese Situation zu verbessern, wurde ein bereits zu Testzwecken während der GUI-Entwicklung genutzter Simulator in das Userinterface integriert, mit dessen Hilfe gefahrlos simulierte Fakten in das System eingegeben werden können. Diese tragen eine entsprechende Kennzeichnung, so dass die dadurch ausgelösten Aktivierungen klar von den durch reale Fakten ausgelösten unterschieden werden können.

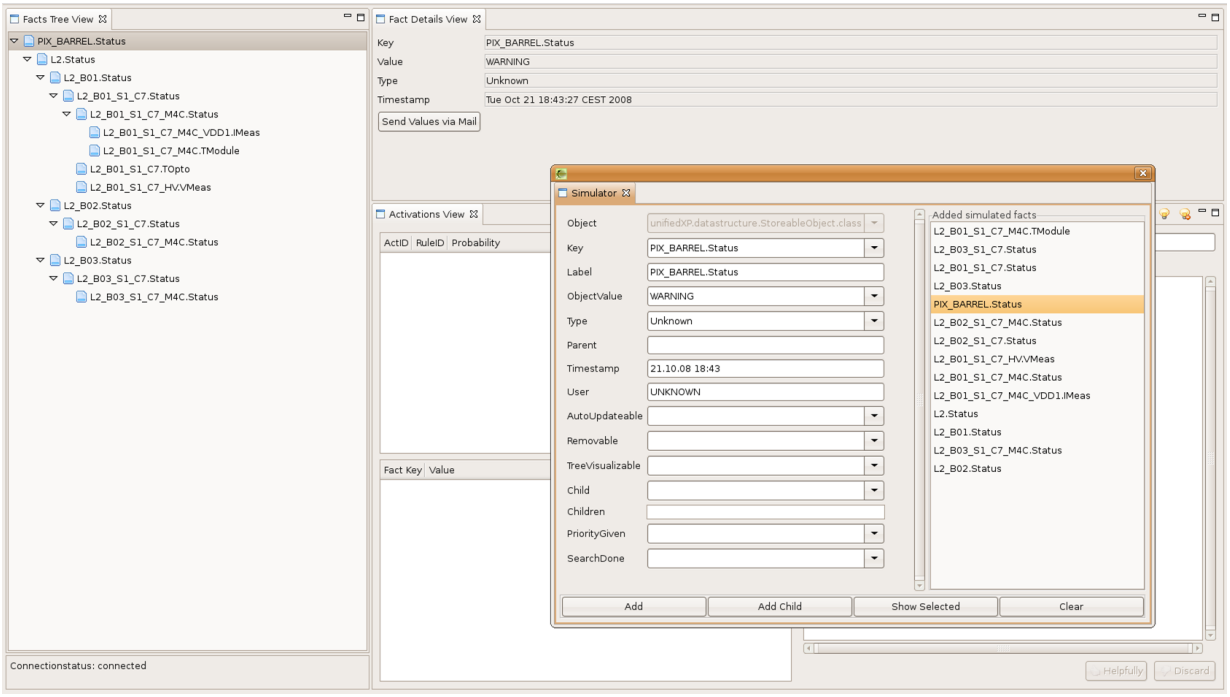


Abbildung 9.9: *Bildschirmfoto der Faktenansicht mit geöffnetem Simulator-Fenster. Der Simulator erlaubt das Erzeugen neuer Knoten sowie die Manipulation aller Felder eines Knotens.*

Abbildung 9.9 zeigt die Faktenansicht mit einem geöffneten Simulatorfenster. Da die Fakten zwischen Server und Client mit Hilfe der in Abschnitt 8.5 erwähnten Container-Objekte ausgetauscht werden, kann es sich bei den Fakten selbst um Instanzen verschiedener Java-Klassen handeln. Insbesondere die Anzahl, Namen und Typen der Objektvariablen eines Faktenobjekts können sich von Einsatzgebiet zu Einsatzgebiet unterscheiden. Um trotzdem zu gewährleisten, dass der Simulator die verschiedenen vorkommenden Faktentypen inklusive ihrer jeweiligen Objektvariablen instanziieren kann, nutzt der Simulator das Prinzip der Java-"Reflektion". Diese ermöglicht es dem Simulator zur Laufzeit, ein Faktum, das ja innerhalb des Clients lediglich in Form der Containerklasse vorliegt, auf seine tatsächliche Klassenzugehörigkeit zu prüfen. Nachdem diese bekannt ist, werden dann mit Hilfe der Reflektion für jede Objektvariable des Faktums entsprechende Eingabefelder dynamisch angelegt. Das in der Abbildung 9.9 gezeigte Simulatorfenster zeigt beispielsweise die Manipulation des Faktums "PIX_BARREL.Status". Dieser Typ besitzt zahlreiche Objektvariablen, die sich in den Eingabefeldern "Key" bis "SearchDone" widerspiegeln. Sollte die DAQ oder andere Subdetektoren andere Objektklassen benötigen, ist der Simulator in der Lage, auch diese zu nutzen.

Neben der Möglichkeit einzelne Fakten per Hand zu manipulieren bzw. zu erzeugen, erlaubt der Simulator auch die automatische Erzeugung aller Kinder eines gegebenen Faktenknotens. Damit ist es in kurzer Zeit möglich, alle relevanten Faktenobjekte für die Simulation eines bestimmten Detektorverhaltens zu erzeugen. Um simulierte Fakten und

daraus resultierende Aktivierungen im Faktenbaum von den eventuell parallel existierenden realen Fakten und Aktivierungen unterscheiden zu können, werden simulierte Fakten durch blaue Balken in den Icons der Baumknoten gekennzeichnet.

9.2 Ideen und Ausblick

Design und Implementation des bisher beschriebenen Expertensystems für das Pixel-DCS erfüllen alle in Kapitel 6 genannten Anforderungen. Der Pixel-Advisor ist damit ein komplett funktionsfähiges Expertensystem, dessen Wissensbasis und damit seine Fähigkeit, gegebene Probleme erfolgreich zu diagnostizieren, durch die kontinuierliche Nutzung am CERN wachsen wird. Durch die im vorherigen Abschnitt angeführten Verbesserungen ist zudem gewährleistet, dass dieser Wissenserwerbsprozess allein von den DCS-Experten, ohne detailliertes Wissen über den Aufbau des Pixel-Advisors selbst, vorangetrieben werden kann.

Einige der möglichen zukünftige Anstrengungen im Bereich der Erweiterung des Pixel-Advisors sollen hier aber dennoch kurz erwähnt werden.

9.2.1 Anpassung der Benutzerschnittstelle

Die Bereitstellung einer möglichst einfach zu bedienenden Benutzerschnittstelle ist eine der wesentlichen Aufgaben dieser Arbeit. Aus Sicht des DCS-Experten ist diese erfüllt, da dieser mit der bestehenden graphischen Benutzeroberfläche alle anfallenden Aufgaben problemlos erledigen kann.

Ob dies allerdings auch für den normalen DCS-Shifter gesagt werden kann, wird sich zeigen, wenn die Software nach ihrer momentanen “Anlernphase” durch die DCS-Experten für den öffentlichen Einsatz freigegeben wird. Es ist anzunehmen, dass einige der implementierten Konzepte noch weitere Vereinfachung benötigen, bevor sie einfach genug handhabbar sind.

Ein absehbares Beispiel einer solchen Bedienungs-Erleichterung für den normalen Nutzer ist der Einsatz der anstehenden neuen JBoss Rules Version 5.0 (der Pixel-Advisor nutzt die zum Zeitpunkt dieser Arbeit aktuelle Version 4.07). Diese wird zahlreiche Verbesserungen im Bereich der unterstützten Regelsyntax mit sich bringen. Dazu gehört beispielsweise die Unterstützung erweiterter DSL-Ausdrücke mit denen es erstmals möglich wird, mehr als eine Variable innerhalb eines DSL-Ausdrucks zu nutzen.

Eine weitere Verbesserung könnte der Einsatz einer datenbankbasierten Regelbasis darstellen, die es ermöglichen würde, die Regeln zentral zu verwalten. Bisherige JBoss Rules Versionen bieten zwar Unterstützung für Regel-Datenbanken, beschränken diese jedoch auf Regeln ohne DSL-Ausdrücke. Da DSL-Ausdrücke mit der anstehenden Version

5.0 ebenfalls in die Datenbank übernommen werden können, ist es prinzipiell möglich, die bisher dateibasierten Regeln in eine entsprechende Datenbank auszulagern, was viele verwaltungstechnische Vorteile mit sich brächte.

Da der im ATLAS-Kontrollraum verfügbare Bildschirmplatz recht begrenzt ist (momentan stehen für Pixel-DCS lediglich zwei Monitore zur Verfügung: einer für die FSM und einer für die Anzeige möglicher PVSS-Alarme), wäre es wünschenswert, die Clients des Pixel-Advisors auf den Laptops der DCS-Shifter zu nutzen. Die Voraussetzung hierzu ist die konsequente Nutzung des bereits implementierten Verschlüsselungsprotokolls SSL. Dieses ermöglicht es prinzipiell, eine sichere Kommunikation zwischen dem im ATLAS-PIT befindlichen Server des Pixel-Advisors und dem dann auf den Shifter Laptops (außerhalb des privaten ATLAS-Netzwerks) laufenden Clients aufzubauen. Der einzige Grund, warum diese Verbindung nicht bereits genutzt wird, liegt in der Tatsache begründet, dass die offizielle ATLAS-IT-Sicherheispolitik den Zugang zu jeglichen Rechnern des ATLAS-Netzes von außerhalb untersagt. Da der eingebaute SSL-Mechanismus bereits für eine sichere Verbindung zwischen den einzelnen GRID-Rechnern sorgt, ist es möglich, dass die Nutzung der Pixel-Advisors-Clients außerhalb des ATLAS-Netzes erlaubt wird.

9.2.2 Verallgemeinerung der Datennahmekomponente

Beim Entwurf und der Umsetzung des Datennahmekonzepts des Pixel-Advisors wurde ein erheblicher Aufwand betrieben, um sicherzustellen, dass der Pixel-Advisor auch von der Pixel-DAQ sowie anderen Subdetektoren genutzt werden kann. Dazu ist auf Grund des Entwurfs der Software lediglich der Austausch der Regelbasis und einiger weniger Datenobjekte innerhalb der Kernkomponente notwendig.

Dementsprechend ist eine Nutzung des Pixel-Advisors, beispielsweise für die DAQ des Pixeldetektors naheliegend und wünschenswert. Bei entsprechender Resonanz ließe sich die Software ebenfalls relativ einfach an die Gegebenheiten anderer Subdetektoren anpassen, was eine ATLAS-zentrale Wartung und Weiterentwicklung des Expertensystems ermöglichen würde. Dies wäre weiterhin ein starkes Argument, den bereits zentral verwalteten DIM-Mechanismus soweit zu verbessern, dass einer der Hauptschwachpunkte des Systems, nämlich das Unvermögen alle Betriebsparameter gleichzeitig zu überwachen, beseitigt werden könnte.

9.2.3 Nutzung historischer Daten

Menschliche Experten greifen bei zahlreichen Problemen auf Vergleiche mit bisherigen Daten zurück[1]. Auch im Bereich des Detektorkontrollsystems sind viele Fälle denkbar, in

[1]Beispiel: “Gestern fuhr der Wagen noch 190 km/h. Heute fährt er nur noch 170 km/h. Eventuell liegt ein Defekt im Einspritzsystem vor.”

denen der Abgleich mit historischen Daten Probleme frühzeitig erkennbar machen könnte, insbesondere dann, wenn es sich um einen schleichenden Prozess, wie etwa die langsame Drift einer Temperatur handelt. Eine entsprechende Regel könnte beispielsweise die zeitliche Änderung der Temperatur überwachen.

Ein weiterer, mindestens ebenso wichtiger Vorteil, der sich durch die Zugriffsmöglichkeit auf historische Daten ergeben würde, liegt in der automatischen Identifikation problematischer Prozessparameter. Man stelle sich dabei vor, dass ein Teil des Detektors aus zu diesem Zeitpunkt unbekannten Gründen in einen problematischen Status wechselt. Mit dem aktuell implementierten Datennahmemechanismus wäre es in so einem Fall zwar möglich, die tatsächlichen Prozessparameter mit den erlaubten Grenzwerten des Soll-Zustands zu vergleichen, kompliziertere Relationen blieben dem System aber verborgen. So ist beispielsweise denkbar, dass lediglich die Modultemperatur jenseits der FSM-Grenzwerte liegt, die eigentliche Ursache für die zu hohe Temperatur aber in einem simultanen Anstieg der Modulspannungen liegt. Sollten diese Spannungen einzeln unterhalb der von der FSM angegebenen Grenzwerte liegen, so würde das Expertensystem diese nicht als problematisch identifizieren. Ein direkter Vergleich, etwa mit der letzten bekannten und lauffähigen Konfiguration des Moduls und der dabei gemessenen Spannungen würde das Problem durch eine sichtbare Abweichung der Modulspannungen von den bisherigen Normalwerten sichtbar machen.

Dies ließe sich innerhalb des Pixel-Advisors nutzen, um halbautomatisch neue Regeln zu erzeugen, sobald ein unbekanntes Problem auf die Abweichung der beteiligten Prozessparameter von ihren historisch bekannten Normalwerten zurückzuführen ist.

Um diese Fähigkeiten in das bestehende Expertensystem einzubauen, müsste dieses mit einer Datenbankschnittstelle versehen werden, um somit Zugriff auf die in der Conditions-DB gespeicherten historischen Daten zu erlangen. Ein weiterer gewichtiger Vorteil einer solchen Datenbankanbindung wäre die Möglichkeit, das Expertensystem quasi “offline” ohne direkte Verbindung zum eigentlichen DCS zu betreiben. Damit würde der bisher bestehende Flaschenhals der DIM-Kommunikation umgangen und bei entsprechender Datenbankanbindung könnten, natürlich mit einem zeitlichen Offset, alle Prozessparameter des Pixeldetektors gleichzeitig vom Expertensystem analysiert werden. Ebenfalls ließen sich neue Regeln sehr realistisch testen, könnte man die Bedingungen die zum erstmaligen Auftreten des Problems führten exakt rekonstruieren. Ein Zugriff auf historische Daten würde dies problemlos dadurch ermöglichen, dass an Stelle der aktuellen Fakten einfach die zum entsprechenden Zeitpunkt vorliegenden und in der Conditions-DB hinterlegten Prozessparameter als Fakten genutzt würden.

9.2.4 Unterstützung von Fuzzy-Logik

An der Fachhochschule Steinfurt beginnen momentan die Planungen einer von RedHat unterstützten Erweiterung der JBoss Rules Expertensystem-Shell. Diese soll die Integra-

tion von “Fuzzy-Logik” in die Regelsyntax und die Inferenzmaschine ermöglichen. Unter dem Stichwort “Fuzzy-Logik” versteht man dabei den Einsatz unscharfer, beschreibender Logik im Gegensatz zu der gebräuchlichen auf den Wahrheitswerten “wahr” und “falsch” beruhenden scharfen Logik.

Fuzzy-Logik vermag unscharfe Begriffe, wie etwa “kalt” oder “heiß”, durch Zugehörigkeitsfunktionen und Wahrscheinlichkeiten auszudrücken. In industriellen Anwendungen werden häufig Fuzzy-Regler eingesetzt, die sich heute beispielsweise in den Belichtungsautomatiken von Spiegelreflexkameras, der Steuerung von Kraftfahrzeug-Antiblockiersystemen und vielen weiteren alltäglichen Technologien wiederfinden.

Die geplante JBoss Rules Erweiterung wird es ermöglichen, unscharfe Regeln der Form: “Wenn die Modultemperatur von vielen Modulen eines Half-Staves deutlich zu hoch ist, dann erhöhe die Kühlleistung stark.” einzusetzen. Dies würde es dem Pixel-Advisor ermöglichen, sprachlich sehr leicht verständliche und dennoch physikalisch (durch die Parametrisierung der Zugehörigkeitsfunktionen) präzise Regeln zu nutzen.

9.3 Erfahrungen mit dem Pixel-Advisor

Mit den am CERN gemachten Erfahrungen lässt sich feststellen, dass das ursprüngliche Konzept eines Expertensystems für das Pixeldetektor-Kontrollsystem erfolgreich umgesetzt werden konnte.

Eine wesentliche Rolle spielte dabei die sorgfältige Auflistung der notwendigen Anforderungen (Kapitel 6), die das Fundament für die Implementation des Pixel-Advisors bilden.

Der ursprüngliche Entwurf der verschiedenen Komponenten und Konzepte erwies sich in weiten Teilen als tragfähig und sinnvoll. Obwohl einige Unzulänglichkeiten erst durch die Nutzung des Expertensystems am CERN aufgedeckt werden konnten, erlaubte das Design des Pixel-Advisors eine schnelle und unproblematische Korrektur durch den Einbau entsprechender Maßnahmen (siehe Abschnitt 9.1).

Der DIM-Mechanismus bietet eine flexible und verlässliche Lösung für die Datennahmekomponente des Expertensystems. Das modulare Konzept des Pixel-Advisors, mit seiner austauschbaren Regeldatenbank, erlaubt im Zusammenspiel mit der CERN-weiten Verbreitung von DIM den Einsatz des Expertensystems auch in anderen Bereichen. Es bleibt allerdings anzumerken, dass der relativ geringe Datendurchsatz des DIM-Mechanismus viele Zugeständnisse im Entwurf des Expertensystems erfordert hat. An dieser Stelle sei erwähnt, dass eine Optimierung der Lastverhaltens von DIM an vielen Stellen wesentlich einfachere und auch wirksamere Strategien zur Auffindung von Detektorproblemen ermöglichen würde (Beispiel: gleichzeitige Überwachung aller Betriebsparameter).

Die eingesetzten Werkzeuge haben, im Rahmen der gestellten Anforderungen, alle Möglichkeiten zur Umsetzung des gewünschten Designs geboten und damit den doch recht hohen Evaluationsaufwand (Kapitel 7) gerechtfertigt. Der Einsatz von Java als gemeinsame Programmiersprache von Datennahmekomponente, Server und Benutzerschnittstelle erwies sich als sinnvoll, da dadurch alle notwendigen Komponenten in einer einzigen Programmiersprache verfasst werden konnten. Eventuell vorhandene Geschwindigkeitsunterschiede zu anderen Programmiersprachen, wie etwa C++, sind im Sinne der Anforderungen nicht relevant, da die DIM-Datennahmekomponente den Flaschenhals des Systems darstellt. Seine Stärken konnte die Kombination aus der Programmiersprache Java und der freien Eclipse-Entwicklungsumgebung während der gesamten Entwurfs- und Umsetzungsphase ausspielen, wobei insbesondere die leistungsfähigen Debugging-Tools der Software eine große Hilfestellung boten.

Eine entscheidende Wahl war sicherlich die Nutzung von JBoss Rules als Expertensystem-Shell. Diese erwies sich nicht nur als sehr gut geeignet, sondern weckte auch das Interesse der Steinfurter Kollegen. Die daraus entstandene Zusammenarbeit führte zu zwei IT-Bachelorarbeiten, in deren Rahmen die Client/Server-Kommunikation sowie die Benutzerschnittstelle des Expertensystems entwickelt wurden. Weitergehende Pläne sehen den Einbau einer Fuzzy-Logik-basierten JBoss Rules-Erweiterung durch die Steinfurter Arbeitsgruppe um Professor Wulff vor, die sich für den Pixel-Advisor als sinnvolle Ergänzung erweisen könnte.

Sicherlich ist die Anzahl der durch den Pixel-Advisor diagnostizier- und behebbaren DCS-Probleme noch zu gering, um von einem “fertigen” Expertensystem zu sprechen. Andererseits war die einfache Erweiterbarkeit der bestehenden Regelbasis von Anfang an einer der fundamentalen Entwurfspfeiler. Daher wurde besonderer Wert darauf gelegt, dass die Nutzer von Anfang an auf eine leistungsfähige und einfach zu bedienende Wissenserwerbskomponente zurückgreifen können, die durch den Einbau des in Abschnitt 8.4.2 erwähnten Regeleditors realisiert wurde.

Momentan besteht die Regelbasis des Pixel-Advisors aus 68 Regeln, von denen 44 den Datenfluss steuern und damit in die Kategorie der in Kapitel 8 vorgestellten Kontrollregeln fallen. Bei den anderen 24 Regeln handelt es sich um die eigentlichen Werteregeln, also das bisher angesammelte Expertenwissen des Systems. Diese bestehenden Regeln sind dazu in der Lage, folgende Probleme zu entdecken und gegebenenfalls Lösungen vorzuschlagen:

- Entdeckung globaler Partitionsprobleme
- Erklärung aller FSM State/Status-Kombinationen auf Modulebene
- Erklärung des “PP0-Undefined”-Zustands im Falle deaktivierter Module
- Entdeckung von Optoboard-Temperaturproblemen
- Entdeckung einer “20 MHz”-Optoboard-Clock

- Entdeckung von “PP2-Trimmer”-Problemen
- Entdeckung “rauschender” Module
- Entdeckung nicht korrekt eingeschalteter Module

Die bisher implementierten Regeln greifen dabei auf bisher 59 DSL-Ausdrücke zurück, während die meisten (57) der angegebenen Regeln weiterhin auch normalen Quellcode gemäß der JBoss Rules Syntax enthalten. Dies liegt sicherlich daran, dass in der momentan genutzten JBoss Rules Version 4.07 viele der notwendigen Bedingungen nur umständlich durch ein oder mehrere DSL-Ausdrücke formuliert werden können und die Nutzung der JBoss Rules Syntax an dieser Stelle bisher flexibler ist. Mit einem höheren Anteil an Nutzern und den Möglichkeiten der neuen JBoss Rules Version kann allerdings davon ausgegangen werden, dass sich die Anzahl der DSL-Ausdrücke schnell erhöht.

Durch intensive Nutzung der Software durch die DCS-Experten und deren Mithilfe sollte die Regelbasis zum Start des LHC-Betriebs im Sommer 2009 in der Lage sein, einen Großteil der bis dahin bekannten Probleme zu erklären.

Eine endgültige Entscheidung über die Nützlichkeit des Expertensystems werden dann die Nutzer fällen.

Zusammenfassung

Im Jahr 2009 wird am europäischen Teilchenphysikzentrum CERN mit dem Large Hadron Collider der bisher leistungsfähigste Hadronenbeschleuniger der Welt in Betrieb genommen. Der ATLAS-Detektor ist eines von vier großen Experimenten an diesem Beschleuniger, die es den beteiligten Physikern erlauben werden, neue Erkenntnisse über den Aufbau der Materie und die Grundkräfte der Natur zu gewinnen.

Als Teil des ATLAS-Experiments spielt der Pixeldetektor eine wichtige Rolle bei der Rekonstruktion der Wechselwirkungsprozesse der kollidierenden Teilchen. Der durchgehende Schichtbetrieb stellt hohe Anforderungen an die Automatisierung des Pixeldetektor-Kontrollsystems DCS, die durch den Einsatz einer Finite State Machine erfüllt werden. Obwohl die FSM dem, vorwiegend aus Nicht-DCS-Experten bestehenden, Betriebspersonal die Kontrolle und Überwachung des Pixeldetektors ermöglicht, waren bisher selbst im Fall bekannter Probleme nur DCS-Experten in der Lage, auftretende Probleme zu lösen.

Im Rahmen dieser Arbeit wurde daher ein Expertensystem “Pixel-Advisor” für das Kontrollsystem des Pixeldetektors entworfen und implementiert. Dieses unterstützt das Bedienungspersonal bei der Diagnose und Behebung möglicher Probleme, die in Zusammenhang mit dem Detektorkontrollsystem stehen und entlastet damit die wenigen verfügbaren DCS-Experten.

Da es sich bei dem zu implementierenden Expertensystem um eine komplette Neuentwicklung handelt, wurde zunächst eine ausführliche Aufstellung der zu erfüllenden Anforderungen an eine solche Software erarbeitet. Nachdem damit der grobe Umriss des zu erstellenden Programms festgelegt war, musste die Frage nach den zu nutzenden Hilfsmitteln beantwortet werden, da nur die Nutzung der bestmöglich geeigneten Methoden und Werkzeuge die Fertigstellung eines Expertensystems im Rahmen dieser Arbeit ermöglichte. Daher wurden die zur Verfügung stehenden Programme einer umfangreichen Evaluiation anhand eines Test-Expertensystems unterzogen, an deren Ende eine fundierte Wahl geeigneter Hilfsmittel getroffen werden konnte. Die resultierende Nutzung der Expertensystem-Shell JBoss Rules in Verbindung mit der Programmiersprache Java und der Entwicklungsumgebung Eclipse erwies sich als sinnvoll und leistungsfähig.

Diese Vorarbeiten ermöglichten die vollständige Umsetzung des Entwurfs in ein funktionsfähiges Expertensystem. Eine besondere Herausforderung stellte die Anbindung

der für ein Expertensystem notwendigen Datennahmekomponente an die Software des Pixeldetektor-Kontrollsystems dar. Mit dem Einsatz des am CERN weitverbreiteten DIM-Protokolls wurde eine flexible, wenn auch, bezüglich ihrer geringen Datenrate, nicht gänzlich unproblematische Datennahme-Lösung gefunden. Weitreichende Maßnahmen zur Kontrolle des Datenflusses wurden daher in das System integriert, um die Stabilität des Detektorbetriebs zu garantieren und gleichzeitig alle zur Lösung eines Problems notwendigen Daten zur Verfügung zu stellen.

Da einer der zentralen Punkte eines Expertensystem seine Fähigkeit zur Aufnahme und Speicherung von Expertenwissen ist, wurde im Rahmen dieser Arbeit eine leistungsfähige Benutzerschnittstelle entwickelt, deren Wissenserwerbskomponente dem Nutzer das einfache Hinzufügen von Wissen in Form von Regeln erlaubt. Weiterhin wurde ein zweistufiges Sicherheitssystem implementiert, das eine geringe Nutzer-Hemmschwelle und eine gleichzeitig hohe Qualität der Regelbasis gewährleistet. Diese Wissenserwerbskomponente ermöglicht es den DCS-Experten nun, Ihr angesammeltes Expertenwissen selbständig in die Regelbasis des Pixel-Advisors einzugeben und für die angestrebte ATLAS-Laufzeit von etwa 10 Jahren zu bewahren.

Die technische Umsetzung von Teilen der Software konnte dabei, Dank der Kooperation mit der Fachhochschule Steinfurt, in zwei, im Rahmen dieser Arbeit betreuter Bachelorarbeiten realisiert werden. Um die langfristige Wartung und damit Nutzung des Expertensystems zu gewährleisten, wurde der Pixel-Advisor zudem zusammen mit dem Grid-Expertensystem "GridXP" auf eine gemeinsame Codebasis, das "UnifiedXP" gestellt.

Seit Abschluss der Entwurfs- und Implementationsphase des Pixel-Advisor-Expertensystems im September 2008 befindet sich das System am CERN im Einsatz. Erste Erfahrungen mit der Software am CERN führten zu einigen Verbesserungen, mit deren Hilfe das Ziel der Bereitstellung eines lernfähigen Expertensystems erreicht werden konnte. Damit das Expertensystem im Sommer 2009 in der Lage ist, **alle** bis dahin bekannten Kontrollsystem-Probleme zu erkennen und entsprechende Lösungsvorschläge zu bieten, müssen mit Hilfe der DCS-Experten allerdings noch zahlreiche neue Regeln in die Regelbasis eingepflegt werden.

Mit dem Pixel-Advisor wurde erstmals eine Software entwickelt, mit deren Hilfe das Betriebspersonal selbst in die Lage versetzt wird, auch kompliziertere Detektorprobleme zu erkennen und zu beheben und damit die DCS-Experten zu entlasten.

Die Nutzung einer gemeinsamen Code-Basis durch das Grid-Expertensystem GridXP und den Pixel-Advisor ermöglicht darüberhinaus die parallele Verbesserung beider Programme. Von zukünftigen Erweiterungen, wie der bereits geplanten Unterstützung von Fuzzy-Regeln auf Basis des UnifiedXP-Codes, werden daher beide Expertensysteme direkt profitieren.

Literaturverzeichnis

[1] Encyclopaedia Brtiannica. Early milestones in ai. http://www.britannica.com/EBchecked/topic/37146/artificial-intelligence, 2008. [Online; accessed 12-September-2008].

[2] ATLAS Collaboration. Atlas detector and physics performance, technical design report, 1999.

[3] ATLAS Collaboration. The atlas experiment at the cern large hadron collider. http://www.iop.org/EJ/toc/1748-0221/3/08, 2008.

[4] ILC Collaboration. International linear collider reference design report. http://www.linearcollider.org/cms/?pid=1000437, 2007.

[5] LHC-Accellerator Collaboration. *LHC Design Report*. http://ab-div.web.cern.ch/ab-div/Publications/LHC-DesignReport.html, 2008.

[6] LHC Computing Grid collaboration. Lhc computing grid webpage. http://lcg.web.cern.ch/LCG/. [Online; accessed 13-September-2008].

[7] Pixel Collaboration. *The Interlock Matrix of the Pixel Detector Control System*, 2008. ATL-IP-ES-0110.

[8] Pixel Collaboration. *Requirements for the Pixel Detector Services*, 2008. ATLAS-IP-ES-0007.

[9] The ATLAS Collaboration. Expected performance of the atlas experiment - detector, trigger and physics, chapter: Higgs boson searches. http://arxiv.org/abs/0901.0512, 2008.

[10] The Pixel Collaboration. *ATLAS Pixel Detector Electronics and Sensors*, 2008.

[11] The ECLIPSE Community. *ECLIPSE Project Homepage*. http://www.eclipse.org/.

[12] RedHat Corporation. *JBoss Rules (Drools) Homepage*. http://www.jboss.org/drools/, 2008.

[13] Daniel Denegri. Standard model physics at the lhc. http://preprints.cern.ch/cgi-bin/setlink?base=AT&categ=Academic_Training&id=AT00000150, 1997. Large Hadron Collider Workshop.

[14] C. Amsler et al. Phys. lett. b667. Technical report, Particle Data Group, 2008.

[15] S. Kersten et al. Towards the final atlas pixel detector control system. In *Topical Workshop on Electronics for Partical Physics, Prague*, 2007.

[16] T. Henss et al. The hardware of the atlas pixel detector control system. In *JINST 2 P05006*, 2007.

[17] Edward A. Feigenbaum. Expert systems: Principles and practice. *The Encyclopedia of Computer Science in Engineering*, 1992.

[18] Edward A. Feigenbaum. A personal view of expert systems: Looking back and looking ahead. Technical report, Department of Computer Science, Stanford University, 1992.

[19] Tobias Flick. *Studies on the Optical Readout for the ATLAS Pixel Detector*. PhD thesis, Bergische Universitaet Wuppertal, WUB-DIS 2006-05 urn:nbn:de:hbz:468-20060600, 2006.

[20] Charles L. Forgy. Rete: a fast algorithm for the many pattern/many object pattern match problem. *Expert systems: a software methodology for modern applications*, pages 324–341, 1990.

[21] Ian Foster. What is the grid? a three point checklist. http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf, 2002. [Online; accessed 12-September-2008].

[22] C. Gaspar. *DIM Distributed Information Management System*. http://dim.web.cern.ch/dim/dim/dim_manual.pdf, 2002.

[23] LEP Electroweak Working Group. Webseite der electroweak working group. http://lepewwg.web.cern.ch/LEPEWWG/, 2008. [Online; accessed 24-September-2008].

[24] ATLAS DCS H. Burckhart et al. *ATLAS DCS Integration Guidelines*. http://edms.cern.ch/file/685451//ATLAS_DCS_Integration_Guidelines.pdf, 2007. ATLAS-DQ-ON-0013.

[25] German Hacker. Grundlagen der teilchenphysik. http://www.solstice.de/grundl_d_tph/titelseite.html, 2003. [Online; accessed 24-September-2008].

[26] Torsten Harenberg. *AMANDA und D0 als Testumgebung fuer das LHC Computing Grid*. PhD thesis, Bergische Universitaet Wuppertal, WUB DIS 2005-09 urn:nbn:de:hbz:468-20050739, 2005.

[27] A. Heister. Search for the standard model higgs boson at lep. Physics Letters B565 (2003) 61, 2003.

[28] ATLAS Collaboration H.H.J. ten Kate. Atlas superconducting magnet system performance. In *Applied Superconductivity Conference, Seattle*, 2006.

[29] Dennis Huning. *Entwicklung einer Rich Client basierten GUI fuer Expertensysteme*. PhD thesis, Fachhochschule Muenster (Steinfurt), 2008.

[30] Frank Iker. *Entwicklung einer Abstraktionsschicht fuer regelbasierte Expertensysteme am CERN Grid-Cluster*. PhD thesis, Fachhochschule Muenster (Steinfurt), 2008.

[31] Martin Imhaeuser. *Anbindung der Detektorkontrolle des ATLAS-Pixeldetektors an das Datennahmesystem und die Conditions-Datenbank sowie erste Studien zur Rekonstruktion von K0s Zerfaellen*. PhD thesis, Bergische Universitaet Wuppertal, WUB DIS 2006-03 urn:nbn:de:hbz:468-20060341, 2006.

[32] John David Jackson. *Klassische Elektrodynamik*. de Gruyter, 2002.

[33] ATLAS Outreach. *Public ATLAS web pages*. http://atlas.ch.

[34] D.H. Perkins. *Introduction to High Energy Physics*. Cambridge University Press, 2000.

[35] Dr. J. Schieck Dr. R. Stroehmer Prof. Dr. S. Bethke, Prof. Dr. O. Biebel. Teilchenphysik mit hoechstenergetischen beschleunigern (tevatron und lhc). Vorlesung WS 2003/04, Max-Planck-Institut fuer Physik (Werner-Heisenberg-Institut), Muenchen.

[36] ATLAS Inner Detector Collaboration S. Haywood. The expected performance of the atlas inner detector. CERN-ATL-COM-PHYS-2008-105.

[37] Pixel DCS S. Kersten. *Overview on the Pixel DCS Hardware*. https://edms.cern.ch/document/707002/, 2008. ATLAS-IP-CD-0002.

[38] Joachim Schultes. *Qualifizierungsmessungen des Spannungsversorgungssystems sowie Konzeptionierung einer Zustandsmaschine fuer die Detektorkontrolle des ATLAS-Pixeldetektors*. PhD thesis, Bergische Universitaet Wuppertal, WUB DIS 2007-04, 2007.

[39] W. Buchmueller und C. Luedeling. *Field theory and the Standard Model*. 2005 European School of High-Energy Physics, 2005.

[40] ATLAS DCS V. Khomoutnikov. *ATLAS DAQ-DCS Communication Software, User's Guide*. http://edms.cern.ch/file/684955/DDC_UG.pdf, 2008.

[41] Sebastian Weber. *Kommunikation zwischen Datennahme- und Kontrollsystem des ATLAS Pixeldetektors*. PhD thesis, Bergische Universitaet Wuppertal, WU D 07-15, 2007.

[42] Wikipedia. Dartmouth conferences — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Dartmouth_Conferences&oldid=235457126, 2008. [Online; accessed 12-September-2008].

Abkürzungsverzeichnis

ALICE ..... **A** **L**arge **I**on **C**ollider **E**xperiment, Seite 21

ATLAS ..... Ursprünglich: **A** **T**oroidal **L**HC **A**paratu**S**. Inzwischen als Eigenname gebraucht., Seite 1

BBIM ..... **B**uilding **B**lock **I**nterlock **M**onitoring, Seite 56

BBM ..... **B**uilding **B**lock **M**onitoring, Seite 58

BOC ..... **B**ack **o**f **C**rate Card, Seite 50

CAN ..... **C**ontroller **A**rea **N**etwork, Seite 52

CERN ..... Aus der ursprünglichen französischen Bezeichnung: **C**onseil **E**uroéen de la **R**echerche **N**ucléaire, Seite 1

CLIPS ..... **C** **L**anguage **I**ntegrated **P**roduction **S**ystem, Seite 88

CMS ..... **C**ompact **M**uon **S**olenoid, Seite 21

CSC ..... **C**athode **S**trip **C**hambers, Seite 37

DAQ ..... **D**ata **Ac**quisition, Seite 67

DB ..... **D**ata **B**ase, Seite 66

DCS ..... **D**etector **C**ontrol **S**ystem, Seite 1

DDC ..... **D**AQ **D**CS **C**ommunication, Seite 67

DIM ..... **D**istributed **I**nformation **M**anagement, Seite 71

DORIC ..... **D**igital **O**pto **R**eceiver **I**ntegrated **C**ircuit, Seite 50

DSL ..... **D**omain **S**pecific **L**anguage, Seite 99

DSS ..... **D**etector **S**afety **S**ystem, Seite 57

ELMB ..... **E**mbedded **L**ocal **M**onitoring **B**oard, Seite 51

EMEC ..... **E**lectro **M**agnetic **E**nd **C**ap, Seite 34

ETM ..... **E**lektro**t**echnik **M**ühlgassner, Seite 61

FCal ..... **F**orward **Cal**orimeter, Seite 34

FDAC ..... **F**eedback **D**igital **A**nalog **C**onverter, Seite 48

FE ..... **F**ront **E**nd, Seite 47

FIT ..... **F**rontend **I**ntegration **T**ool, Seite 64

FPGA ..... **F**ield **P**rogrammable **G**ate **A**rray, Seite 57

FSM ..... **F**inite **S**tate **M**achine, Seite 67

GCS ..... **G**lobal **C**ontrol **S**tation, Seite 69

GDAC ..... **G**lobal **DAC**, Seite 48

GUI ..... **G**raphical **U**ser **I**nterface, Seite 64

HEC ..... **H**adronic **E**nd **C**ap, Seite 35

HLT ..... **H**igh **L**evel **T**rigger, Seite 39

IBM ..... **I**nternational **B**usiness **M**achines Corporation, Seite 94

IDB ..... **I**nterlock **D**istribution **B**ox, Seite 57

IDE ..... **I**ntegrated **D**evelopment **E**nvironment, Seite 94

ILC ..... **I**nternational **L**inear **C**ollider, Seite 14

iseg ..... **I**nstitut für **S**pezial**e**lektronik **G**mbH, Seite 52

JESS ..... **J**ava **E**xpert **S**ystem **S**hell, Seite 88

JNI ..... **J**ava **N**ative **I**nterface, Seite 97

LAr ..... **L**iquid **Ar**gon, Seite 33

LEP ..... **L**arge **E**lektron **P**ositron Collider, Seite 11

LHC ..... **L**arge **H**adron **C**ollider, Seite 1

LHCb ..... **L**arge **H**adron **C**ollider **b**eauty experiment, Seite 21

LINAC ..... **Lin**ear **Ac**cellerator, Seite 23

LISP ..... **Lis**t **P**rocessing, Seite 88

LVDS ..... **L**ow **V**oltage **D**ifferential **S**ignal, Seite 48

MCC ..... **M**odule **C**ontrol **C**hip, Seite 48

MDT ..... **M**onitored **D**rift **T**ube, Seite 37

NTC ..... **N**egative **T**emperature **C**oefficient, Seite 49

OPC ..... **O**bject linking and embedding for **P**rocess **C**ontrol, Seite 51

PCC ..... **P**arallel **C**ooling **C**ircuit, Seite 44

PIN ..... **P**ositive **I**ntrinsic **N**egativ, Seite 50

PP ..... **P**atch **P**anel, Seite 51

PS ..... **P**roton **S**ynchrotron, Seite 23

PVSS ..... **P**rozess **V**isualisierungs und **S**teuerungs **S**oftware, Seite 61

QCD ..... **Q**uantum **C**hromo **D**ynamics, Seite 12

RAID ..... **R**edundant **A**rray of **I**ndependent **D**isks, Seite 40

RGMA ..... **R**elational **G**rid **M**onitoring **A**rchitecture, Seite 121

ROD ..... **R**ead **O**ut **D**river, Seite 50

RoI ..... **R**egion **o**f **I**nterest, Seite 39

RPC ..... **R**emote **P**rocedure **C**all, Seite 71

RPC ..... **R**esistive **P**late **C**hamber, Seite 38

SC-OL ..... **S**upply and **C**ontrol for the **O**pto**l**ink, Seite 55

SCS ..... **S**ubdetector **C**ontrol **S**tation, Seite 69

SCT ..... **S**emi **C**onductor **T**racker, Seite 27

SI ..... **S**ystème **I**nternational d'Unités, Seite 5

SIT ..... **S**ystem **I**ntegration **T**ool, Seite 65

SPS ..... **S**uper **P**roton **S**ynchrotron, Seite 23

SQL ..... **S**tructured **Q**uery **L**anguage, Seite 66

SQL ..... **S**tructured **Q**uery **L**anguage, Seite 100

SSL ..... **S**ecure **S**ockets **L**ayer, Seite 121

TCP/IP ..... **T**ransmission **C**ontrol **P**rotocol / **I**nternet **P**rotocol, Seite 55

TDAC ..... **T**hreshold **DAC**, Seite 48

TGC ..... **T**hin **G**ap **C**hamber, Seite 38

TOT ..... **T**ime **O**ver **T**hreshold, Seite 48

TRT ..... **T**ransition **R**adiation **T**racker, Seite 27

VCSEL ..... **V**ertical **C**avity **S**urface **E**mitting **L**aser, Seite 50

VDC ..... **V**CSEL **D**river **C**hip, Seite 50

VMEbus ..... **V**ersa **M**odule **E**urocard **bus**, Seite 59

W-Ie-Ne-R ... **W**erk für **I**ndustrie**e**lektronik - **N**uklear**e**lektronik - **R**egelungstechnik, Seite 55

XML ..... E**x**tensible **M**arkup **L**anguage, Seite 65

Abbildungsverzeichnis

Tabellenverzeichnis

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Diese Dissertation ist keinem weiteren Fachbereich einer wissenschaftlichen Hochschule vorgelegt worden.

Wuppertal, den 1. Februar 2009

(Tobias Henß)