

**Weiterentwicklung von
Methoden zur
interaktiven Modellierung
und zur Visualisierung in
ATLAS-GRAMOVIS**

Weiterentwicklung von Methoden zur interaktiven Modellierung und zur Visualisierung in ATLAS-GRAMOVIS

Francesco Cester
Matthias Küntzel
Uladimir Samadurau
Josef Scheuer
Thomas Voggenberger

Dezember 2018

Anmerkung:

Das diesem Bericht zugrunde liegende Forschungsvorhaben wurde mit Mitteln des Bundesministeriums für Wirtschaft, und Energie (BMWi) unter dem Kennzeichen RS1537 durchgeführt.

Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Auftragnehmer.

Der Bericht gibt die Auffassung und Meinung des Auftragnehmers wieder und muss nicht mit der Meinung des Auftraggebers übereinstimmen.

Deskriptoren

AGM, AIG, ATHLET, ATLAS, ATM, Eingabedaten, GRAMOVIS, GUI, Reaktorsicherheit, Simulation, Visualisierung

Kurzfassung

Das Gesamtziel des Vorhabens war die Weiterentwicklung der Simulationsumgebung ATLAS-GRAMOVIS für Rechenprogrammsysteme zur Reaktorsicherheitsanalyse. Diese Simulationsumgebung stellt Methoden zur grafischen Modellierung und Ergebnisauswertung für die Simulationsmodelle bereit. Unter den Begriffen grafische Modellierung und grafische Ergebnisauswertung werden computergestützte Methoden des Prä- und Postprocessings der Simulationsmodelle zusammengefasst, die dem Anwender bei der Durchführung der Simulation Hilfestellung leisten können. Dafür eignen sich grafische Eingabewerkzeuge, interaktive Bedienoberflächen zur Durchführung der Simulationsrechnung sowie die visuelle Darstellung der Simulationsergebnisse. Ein besonderer Schwerpunkt der Entwicklung waren die durch den Systemcode ATHLET gestellten Anforderungen.

Für die interaktive Eingabeerstellung von ATHLET werden im ATHLET Input Modeller Module für thermohydraulische und leittechnische Systeme bereitgestellt. Mit dem erreichten Entwicklungsstand ist der ATHLET Thermohydraulic Modeller ATM in der Lage, einen großen Teil der ATHLET Eingabe für Fluidsysteme zu unterstützen. Mit den parametrischen Modellierungsansätzen können die erstellten Daten effizient angepasst werden. Für den bereits in der Anwendung genutzten ATHLET GCSM Modeller wurden Verbesserungen durchgeführt. Mit der Bereitstellung einer Linux-Version von AGM ist ein Schritt in Richtung Plattformunabhängigkeit gelungen. Insgesamt kann die AGM Entwicklung als ausgereift bezeichnet werden.

Der Analysesimulator ATLAS zur Visualisierung der Simulation und die ATHLET Input Graphic AIG für die Eingabekontrolle wurden mit einem umfangreichen Reengineering in weiten Teilen neu erstellt. Mit der Nutzung der GUI-Bibliothek Qt wurde eine plattformunabhängige Implementierung der Software möglich. Eine weitgehende Modularisierung der verschiedenen Funktionen in Bibliotheken stellt sicher, dass die Programme langfristig wartbar und erweiterungsfähig sind. ATLASneo enthält mit den Modulen zur Steuerung von ATHLET sowie der Visualisierung von Daten in Tabellen, Charts und dynamischen Grafiken bereits einen großen Teil der in der Anwendung benötigten Funktionalität. Die neue AIG verfügt neben allen bisher vorhanden Funktionen über erweiterte Möglichkeiten in der Bedienung und bei der Darstellung der Daten

Abstract

The overall objective of the project was the further development of the ATLAS-GRAMOVIS simulation environment for reactor safety analysis codes. This simulation environment provides methods for graphical modelling and the evaluation of results for the simulation models. Under the terms graphical modelling and graphical evaluation of results, computer-aided methods of pre- and post-processing for the simulation models are summarized, which support the user in the different steps of a simulation. Graphical input tools, interactive user interfaces for performing simulation runs as well as the visual representation of the simulation results are suitable means for these tasks. A particular focus in the development was imposed by the requirements of the system code ATHLET.

For the interactive creation of ATHLET input data the ATHLET Input Modeller provides two modules for thermohydraulic and process control systems. The ATHLET Thermohydraulic Modeller ATM is ready for practical use and currently supports a large part of the ATHLET input for fluid systems. With the parametric modelling approaches, the created data can be adapted efficiently. Improvements have been made to the ATHLET GCSM Modeller that has been already applied for real systems. The deployment of a Linux version of AGM has taken a step towards platform independence. Overall, AGM is fully operative and the development is generally completed.

The analysis simulator ATLAS for visualization of the simulation and the ATHLET Input Graphic AIG for the input control were reimplemented in large parts with a comprehensive reengineering. With the use of the GUI library Qt, a cross-platform implementation of the software became possible. An extensive modularization of the various functions in libraries ensures that the programs are long-term maintainable and expandable. ATLASneo already contains a large part of the necessary functionality found in ATLAS. Based on new data interface design modules for interactive control of ATHLET simulations and the visualization of data in tables, charts and dynamic graphics have been developed. The new AIG provides not only all the functions of the recent design, but goes beyond by extended capabilities in the operation and the display of data.

Inhaltsverzeichnis

	Kurzfassung.....	I
	Abstract.....	III
1	Einleitung	1
2	Zielsetzung und Arbeitsprogramm.....	3
2.1.1	Werkzeuge zur Generierung von Daten für ATHLET Modelle	3
2.1.2	Reengineering der ATLAS Software	4
2.1.3	Reengineering der ATHLET Input Graphics.....	5
3	Stand der Wissenschaft und Technik	7
4	Arbeiten und Ergebnisse	11
4.1	Werkzeuge zur Generierung von Daten für ATHLET Modelle	11
4.1.1	Basissoftware für ATHLET Eingabewerkzeuge.....	11
4.1.2	Modellierung und Darstellung von Objektnetzen	12
4.1.3	Datenimport für ATM	35
4.1.4	Erzeugung und Test der Eingabedaten.....	40
4.1.5	Modellierung parametrischer Objekte	42
4.1.6	Datenexport für Visualisierung	47
4.1.7	Qualitätssicherung und Dokumentation	51
4.1.8	Erweiterung der Leittechnikmodellierung mit AGM.....	54
4.2	Reengineering von ATLAS	61
4.2.1	Anwendungsfälle von ATLAS.....	61
4.2.2	ATLASneo	64
4.2.3	Erzielte Ergebnisse.....	68
4.3	Reengineering der ATHLET Input Graphic AIG	91
4.3.1	Trennung der Programmteile	91
4.3.2	Implementierung einer GR-File und SVG-Parser-Klasse	92
4.3.3	Baumdarstellung der Objekte.....	95

4.3.4	Verbesserung der Bedienung	98
5	Zusammenfassung und Ausblick	103
5.1	ATHLET Input Modeller	103
5.2	ATLASneo	104
5.3	ATHLET Input Graphic.....	105
	Literaturverzeichnis.....	107
	Abbildungsverzeichnis.....	111
	Tabellenverzeichnis.....	115
	Abkürzungsverzeichnis.....	117
A	Anhang: Berechnung von Formverlustkoeffizienten in ATM	119
A.1	Rohrbogen.....	119
A.2	Kniestück	120
A.3	Diffusor	121
A.4	Düse (Querschnittsverengung)	123
B	Anhang: Zusätzliche Hinweise zur Konvertierung von Leittechnikmodellen	125
B.1	Verwendung des Konvertierungsprogramms	125
B.2	Abweichungen bei der Konvertierung	125
B.3	Verwendung des Sortierprogramms	126
C	Anhang: Ergebnisse der Umfrage zu ATLAS Anwendungsfällen	127

1 Einleitung

Die Komplexität von Rechenprogrammsystemen zur Simulation von Stör- und Unfallabläufen in KKW hat sich in den letzten Jahren beispielsweise durch die Implementierung neuer, mehrdimensionaler Modelle ständig erhöht. Mit letzteren sind auch die Anforderungen an die Anwender gestiegen. Die Erstellung von Eingabedaten für die Simulationscodes, die Auswertung der Ergebnisse mit hoher räumlicher und zeitlicher Auflösung sowie das Verständnis der aufgetretenen Phänomene sind ohne zusätzliche Werkzeuge kaum möglich. Es werden fortschrittliche, computergestützte Methoden benötigt, die den Codeanwender bei der Vorbereitung, der Durchführung und Auswertung der Analysen unterstützen. Diese Methoden tragen dazu bei, Fehler bei der Datenerstellung zu vermeiden und das Verständnis der Ergebnisse von Simulationen zu verbessern. Die Methoden wurden im Vorhaben RS1537 weiterentwickelt. Der Bericht enthält eine detaillierte Beschreibung der wesentlichen Arbeiten und Ergebnisse des Vorhabens zur Erweiterung von ATLAS-GRAMOVIS.

2 Zielsetzung und Arbeitsprogramm

Das Gesamtziel des Vorhabens besteht darin, für Simulationsprogramme der Reaktorsicherheit eine interaktive grafische Computerplattform für die Erstellung der Eingabedaten für die Modelle und für die Ergebnisauswertung weiter zu entwickeln. Die Plattform umfasst CAD-basierte Werkzeuge zur Vorgabe von Geometrie und Gitterdaten, grafische Verfahren zur Nachbildung thermohydraulischer und leittechnischer Systeme, Prozeduren zur Erzeugung der modellspezifischen Eingabeformate, Bedienoberflächen zur interaktiven Durchführung der Simulationsrechnungen sowie die grafische Aufbereitung und visuelle Darstellung der Simulationsergebnisse. Ein besonderer Schwerpunkt der Erweiterungen sind die durch den Systemcode ATHLET gestellten Anforderungen bei allen Phasen einer Störfallsimulation.

Das Vorhaben setzt inhaltlich auf dem durch das BMWi geförderten Vorhaben RS1199 „Methodenentwicklung zur Modellierung technischer Systeme und zur Ergebnisauswertung für Simulationsprogramme“ /VOG 15/ und der langjährigen Entwicklung des Analysesimulators ATLAS auf. Die dort entwickelten Methoden zur grafischen Modellierung und Visualisierung GRAMOVIS eignen sich generell für unterschiedliche Simulationscodes, müssen aber, entsprechend der codespezifischen Anforderungen, in den Details angepasst und erweitert werden. Im beschriebenen Vorhaben werden dafür die drei folgenden Schwerpunkte gesetzt.

2.1.1 Werkzeuge zur Generierung von Daten für ATHLET Modelle

Die Erstellung der Eingabedaten für die verschiedenen Teilmodelle von ATHLET soll mit interaktiven Methoden ermöglicht werden. Dazu wird der "ATHLET Thermohydraulic Modeller" (ATM) weiterentwickelt, insbesondere im Hinblick auf die Unterstützung weiterer Modelle aus ATHLET, wie beispielsweise Wärmeleitobjekte, die Geometriedarstellung der Fluidobjekte und die Modellierung von Objekten und Komponenten mit Parametern.

Das bereits eingesetzte Werkzeug zur Leittechnikmodellierung, der "ATHLET GCSM Modeller" (AGM) wird ergänzt um eine Importfunktion von bestehenden Leittechnikmodellen und die Verfügbarkeit unter Linux.

Die Umsetzung erfolgt durch die folgenden Arbeitsschwerpunkte:

- Erweiterung des ATM-Prototyps (Diskretisierung, Geometriedarstellung, Wärmeleitobjekte)
- Modellierung von Sondermodellen (Druckhalter, Dampferzeuger, Neutronenkinetik)
- Erzeugung eines vollständigen ATHLET Eingabedatensatzes
- Datenimport für ATM (komplette Datensätze, Änderungen)
- Modellierung parametrischer Objekte
- Datenvisualisierung (Exportfunktion für ATLAS, ATM intern)
- Qualitätssicherung, Benutzerhandbuch und Systemtest
- Erweiterung der Leittechnikmodellierung mit AGM (Linux-Version, Datenimport)

2.1.2 Reengineering der ATLAS Software

Der Analysesimulator ATLAS dient zur interaktiven Durchführung einer Simulation und zur Visualisierung der Ergebnisse. Die Visualisierungsmöglichkeiten sind sehr umfangreich, beispielsweise X/Y-Diagramme, Prozessbilder und spezielle, automatisiert erstellbare Diagramme. Neben der Ergebnisvisualisierung kann die Simulation interaktiv gesteuert werden, z. B. Operateureingriffe oder Komponentenausfälle.

Für den Analysesimulator ATLAS wird ein komplettes Reengineering durchgeführt. Die Softwarequalität soll erhöht werden, eine neue Benutzeroberfläche (GUI) unter Nutzung quelloffener Software entwickelt und damit die Bedienbarkeit des Programms verbessert werden. Die Lauffähigkeit unter Linux soll erreicht werden. Diese Einzelziele können durch eine Reihe von Maßnahmen erreicht werden, wie Refactoring und Neuerstellung des Programmcodes, eine neue GUI-Bibliothek, ein standardisiertes Bildformat und entsprechende Programmdokumentation.

Die Umsetzung erfolgt durch die folgenden Arbeitsschwerpunkte:

- Modularisierung und Refactoring (Neuprogrammierung, Programmbibliothek, Dokumentation)
- Programmmodul für X/Y-Diagramme
- Einführung standardisierter Tests
- Ersetzen der GUI-Bibliothek MFC
- Weiterentwicklung des ATLAS Bildformats (OpenSource, Bilddynamik, Bildkonvertierung)
- Weiterentwicklung von Funktionen in ATLAS

2.1.3 Reengineering der ATHLET Input Graphics

Die "ATHLET Input Graphics" (AIG) ist ein spezielles Werkzeug zur Visualisierung und Kontrolle der ATHLET Eingabedaten für "Thermo-Fluid Objects" (TFO) und "Heat-Conducting Objects" (HCO). Im Wesentlichen können deren geometrischen Daten, Verbindungen und ihre Nodalisierung automatisiert dargestellt werden. Die wesentliche Funktion des Werkzeugs ist die visuelle Überprüfung der korrekten geometrischen Abbildung eines thermohydraulischen Systems. Zusätzlich besteht die Möglichkeit, die Darstellungen von AIG nach ATLAS zu exportieren und dort die lokalen dynamischen Daten in den Geometrieobjekten zu visualisieren. Für die AIG soll ebenfalls ein Reengineering durchgeführt werden, die eine neue, plattformunabhängige und besser bedienbare Benutzeroberfläche verfügbar macht.

Die Umsetzung erfolgt durch die folgenden Arbeitsschwerpunkte:

- Modularisierung, Refactoring und Kommentierung
- Neue GUI-Bibliothek
- Verbesserung der Bedienung
- Anpassung an die ATHLET-Entwicklung

3 Stand der Wissenschaft und Technik

In diesem Abschnitt wird der Stand der Technik zu Beginn der Arbeiten im Bereich der vorhandenen Methoden zur Erzeugung der Daten für die Simulationsmodelle beschrieben. Zum Vergleich werden die Werkzeuge zum Präprozessing vergleichbarer Simulationsmodelle für Reaktorsicherheitsanalysen gegenübergestellt. Für ATLAS und AIG wird aufgezeigt, wo der Zustand der Software nicht mehr auf dem aktuellen Stand bei der Softwaretechnik ist.

Basis für die Simulation einer Anlage mit ATHLET-Modellen ist stets ein ASCII-Eingabedatensatz, der den Vorgaben der Eingabedatenbeschreibung entsprechen muss. Die Eingabedaten umfassen im Wesentlichen die Eigenschaften der Modellkomponenten (Objekte) wie Geometrie, Materialien, Kennwerte und die Verbindungen der verschiedenen Objekte untereinander. Die zunehmende Simulationstiefe, eine hohe örtliche Auflösung, Redundanzen und detaillierte Leittechnikmodellierung, führen zu sehr großen Datensätzen mit extrem vielen Einzelobjekten.

Für das Leittechniksimulationsmodell GCSM /AUS 12/ von ATHLET ist seit langer Zeit ein grafischer Editor im Einsatz, der zur Nachbildung der leittechnischen Systeme im Rahmen der Erstellung von Anlagensimulatoren genutzt wird. Der GCSM-Eingabegenerator /JAK 97/ ist mit der Expertensystemsoftware „G2“ von GENSYM realisiert. Dies ist ein umfangreiches Softwarewerkzeug, z. B. für Prozessüberwachung, das entsprechende Lizenzkosten erfordert und deshalb nicht allgemein verfügbar ist.

Im Vorhaben RS1199 wurde mit der Entwicklung interaktiver, grafischer Werkzeuge begonnen, die den Prozess der Eingabeerstellung unterstützen. Für die Modellerstellung von komplexen leittechnischen oder thermohydraulischen Reaktorsystemen wurde ein generell verwendbarer grafischer Netzwerkkeditor entwickelt. Dieser ermöglicht die Nachbildung der Topologie der Netze durch das interaktive Zusammenschalten von Objekten. Der interaktive Editor umfasst eine erweiterbare Objektbibliothek mit Verbindungselementen, einen Objektbrowser, Eingabemasken für Objektdaten und den Datenexport für das Simulationsmodell. Für die Leittechnikmodellierung in ATHLET wurde auf dieser Basis der „ATHLET GCSM Modeller“ (AGM) implementiert. Die Modellierung eines leittechnischen Systems erfolgt durch die Verschaltung einzelner Blöcke und die Vorgabe der Parameter. Eine interne Systemsimulation in AGM ist durch eine gekoppelte Simulationsbibliothek möglich. Aus den in AGM vorhandenen Daten können die für ATHLET-GCSM lesbaren ASCII-Eingabedateien erzeugt werden. Die Entwicklung des

Leittechnikmodellierers AGM hat einen Stand erreicht, der den Einsatz in Anwendungen zur Simulation umfangreicher Systeme erlaubt. Auf der gleichen Basis wie AGM wurde die Entwicklung eines Prototyps für die Erzeugung von thermohydraulischen Netzen, der „ATHLET Thermohydraulic Modeller“ (ATM) begonnen, der nur über sehr beschränkte Funktionen verfügt.

Im internationalen Bereich gibt es vergleichbare Werkzeuge zur Erstellung der Eingabedaten. Für TRACE, ein Thermohydraulikcode der US-NRC auf Basis von TRAC und RELAP5 ist SNAP, „Symbolic Nuclear Analysis Package“ /APT 12/ verfügbar, das für die Erstellung der Eingabedaten, Ausführung und Auswertung der Analysen verwendet werden kann. SNAP ist als generelles Werkzeug konzipiert und unterstützt neben TRACE auch noch andere Codes wie CONTAIN, COBRA, MELCOR, PARCS und RELAP5 und damit die wesentlichen US-Codes für Reaktorsicherheitsanalysen. Die wesentlichen Elemente von SNAP sind der „Model Editor“ und der „Navigator“. Im Editor können einzelne Komponenten des Modells, z. B. Fills, Pipes, Pumps usw. grafisch in einem Netzwerk zu einem Anlagenmodell zusammengesetzt werden. Im Navigator werden die erzeugten Komponenten in einer Baumstruktur angezeigt und die Dialoge zur Eingabe der Komponentendaten können aufgerufen werden. Neben den physikalischen Eingabedaten können auch sogenannte „Views“ der Komponenten definiert werden, die für die Visualisierung der dynamischen Daten und zur interaktiven Steuerung verwendet werden. Die Möglichkeiten sind ähnlich wie die in ATLAS vorhandenen. Zur Ausführung von Rechnungen enthält SNAP zusätzlich Werkzeuge zum Start auf einem Netzwerkcomputer und zur Kontrolle der Rechnung während der Laufzeit. Für den Integralcode ASTEC wird von IRSN gegenwärtig der „XASTEC Data Set Editor“ zur interaktiven Generierung der Eingabedaten entwickelt. Die Entwicklung ist noch in einem frühen Stadium und nutzt ähnliche Methoden wie in SNAP und AGM/ATM. Alle wichtigen Module von ASTEC sollen unterstützt werden. XASTEC ist in Java implementiert und damit plattformübergreifend verfügbar. Bestehende ASTEC Datensätze sollen importiert werden können. Von XASTEC aus können ASTEC-Rechnungen, die Ergebnisvisualisierung und ein Werkzeug zur Unsicherheitsanalysen gestartet werden.

Die Entwicklung der Softwareprodukte ATLAS und AIG erstreckt sich über einen Zeitraum von mehr als 20 Jahren. Der vorhandene Funktionsumfang für das Postprocessing von ATHLET aber auch anderer RS-Codes ist sehr groß und deckt die wichtigsten Anforderungen ab. Im Gegensatz dazu entspricht die Implementierung der Programme

nicht mehr dem Stand der aktuellen Computertechnik. Der Programmcode ist durch das kontinuierliche Hinzufügen neuer Funktionen zunehmend unübersichtlich und in den Details schwer verständlich geworden, auch wegen der unzureichenden Dokumentation der Erweiterungen. Dadurch sind die Programmpflege und das Hinzufügen neuer Funktionalität, besonders für neues Personal, nur mit großem Aufwand durchführbar, auch wegen der häufigen Seiteneffekte der Programmänderungen auf andere Teile des Codes. Optimierungsbedarf bei AIG besteht in der Bedienbarkeit mit handgeführten Eingabegeräten (Maus, Trackball, etc.) z. B. beim Verschieben von grafischen Objekten. Das Programm bietet gegenüber modernen GUIs noch ein großes Verbesserungspotential.

ATLAS und AIG greifen u. a. auf externe Programmbibliotheken wie die Microsoft Foundation Class Library /MFC 14/ und OpenGL /OGL 14/ zu. Diese Bibliotheken sind auf die aktuellen WINDOWS-Versionen abgestimmt und damit fortlaufenden Änderungen unterworfen. Die enge Verzahnung mit der Applikationslogik hat in der Vergangenheit mehrfach dazu geführt, dass bei einer Neugenerierung der AIG oder von ATLAS mit aktuellen externen Bibliotheken Fehler auftraten, die die Nutzbarkeit der Programme erheblich einschränken und die sich nicht oder nur mit sehr großem Aufwand beheben lassen.

4 Arbeiten und Ergebnisse

4.1 Werkzeuge zur Generierung von Daten für ATHLET Modelle

In diesem Arbeitspaket wurden die interaktiven Werkzeuge zur Erstellung der ATHLET Eingabe weiterentwickelt. Zur Modellierung von Systemen mit Thermo-Fluidobjekten (TFO) und deren Netzwerktopologie wurde der „ATHLET Thermohydraulic Modeller“ (ATM) für einen weiten Funktionsumfang entwickelt. Zur Nachbildung der Reaktorleittechnik und von Hilfssystemen steht der „ATHLET GCSM Modeller“ (AGM) zur Verfügung, der bereits für den produktiven Einsatz geeignet ist und durch Erweiterungen ausgebaut wurde.

4.1.1 Basissoftware für ATHLET Eingabewerkzeuge

Zur Realisierung von ATM und AGM wird als Basissoftware „SimInTech“ /SIT 15/ verwendet, ein generelles System zur Modellierung von Netzen aus einzelnen Objekten. Dieses System ist z. B. mit „Simulink“ vergleichbar, eine weit verbreitete blockorientierte Entwicklungsumgebung für die Mehrdomänen-Simulation und Model-Based Design. Mit dem Erwerb des Quellcodes von SimInTech besteht das Recht, damit erstellte Anwendungen „runtime free“, also ohne zusätzliche Gebühren, an andere Anwender weiterzugeben. Dies ist eine wesentliche Anforderung für alle ATHLET-Softwarebestandteile. Die Basissoftware kann durch „Plugins“ modular erweitert werden und stellt dafür grundlegender Funktionen bereit:

- Erweiterbare Bausteinbibliothek
- Zusammenschaltung der Elemente mit Verbindungen zwischen Ein- und Ausgängen
- Eingabemasken für die Objekt-Parameter
- Testmöglichkeit der Systeme
- Erzeugung codespezifischer Eingabe oder von Quellcodeprogrammen

Diese Funktionen sind teilweise direkt ohne weitere Programmierung nutzbar oder können über eine umfangreiche Schnittstellenbibliothek im Programmcode der Plugins angesprochen werden. Prinzipiell sind aber auch Anpassungen der Basissoftware unabhängig vom Hersteller möglich, so dass prinzipiell beliebige Erweiterungen in der

Funktionalität möglich sind. Weitere Details zur Basissoftware sind in /VOG 15/ beschrieben.

4.1.2 Modellierung und Darstellung von Objektnetzen

Die Modellierung für ATHLET umfasst insbesondere die Nachbildung von Rohrleitungen und wärmeleitenden Strukturen. ATHLET stellt dazu Thermo-Fluidobjekten (TFO) und Wärmeleitungsobjekte (HCO) bereit, die entsprechend den realen Gegebenheiten zu den Anlagensystemen in Netzen verbunden werden.

4.1.2.1 Modellierung eines Thermo-Fluidobjekts (TFO)

Eine einfache Modellierung eines Thermo-Fluidobjekts (TFO) war im Prototyp von ATM aus RS1199 bereits realisiert. Die Funktionalität wurde von ATHLET-Entwicklern und Anwendern der GRS getestet. Die Probleme und Erkenntnisse wurden ausgewertet und diskutiert. Generell gilt die Anforderung, dass eine geometrietreue Darstellung einzelner Rohrleitungselemente

- Gerades Rohrstück
- Diffusor
- Düse
- Rohrbogen

in hoher Detaillierung benötigt wird. Ähnlich wie in der vorhandenen ATHLET Input Graphic AIG sollen beispielsweise Raumkoordinaten, geodätische Höhen und Strömungsquerschnitte grafisch korrekt dargestellt werden.

Als besonders nachteilig wurde die sehr einfache Darstellung und Modellierung der Bogenelemente in den TFO und eine fehlende Unterstützung bei der Berechnung von Druckverlusten (form loss coefficients) bemerkt.

Die ursprünglich geplante Darstellung der Nodalisierung, d. h. die Einteilung eines TFO in diskrete Volumenelemente, in einer schematischen Form, wurde als unzureichend angesehen. Vielmehr ist zur Beurteilung einer angemessenen Vorgabe die Lage der Nodengrenzen in der geometrischen Darstellung des TFO an der korrekten Position anzuzeigen.

Die Berücksichtigung dieser Anforderungen erforderte die vollständige Überarbeitung und eine teilweise Neuerstellung der Programme im Prototyp von ATM zur Modellierung eines TFO.

Zur Modellierung eines TFO stellt ATM nun über ein Objektauswahlmenü (Abb. 4.1) eine Reihe von Basiselementen zur Verfügung.

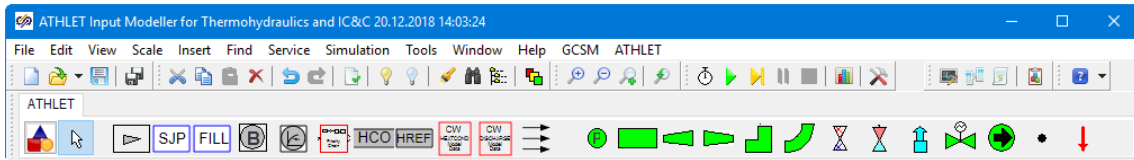


Abb. 4.1 Menüleiste in ATM zur Erstellung von TFO

Die gegenwärtig verfügbaren Fluidobjekte sind in Abb. 4.2 dargestellt:







ATHLET Pipe	
ATHLET Single Junction Pipe	
ATHLET Fill	
ATHLET Branch	
ATHLET Time Dep. Volume	
ATHLET Cross Connection	

Abb. 4.2 Gegenwärtig verfügbare Fluidobjekte

Die Fluidobjekte können eine allgemeine Rohrleitung oder spezielle Varianten beschreiben. Ein Branch kann in ATHLET nicht weiter in einzelne Volumenelemente unterteilt werden und kann keine Junctions enthalten. Die TFO besitzen eine Geometriebeschreibung und müssen in ATM durch die Verbindung von hydraulischen Basiselementen (siehe Abb. 4.3) definiert werden.




Gerades Rohrstück	
Diffusor	
Düse	
Rohrbogen (einfach, 90 Grad)	
Rohrbogen (Segmentzahl u. Winkel bel.)	
Anfangspunkt für TFO (Koordinaten)	

Abb. 4.3 Geometrische Basiselemente für ein TFO

Zur Beschreibung eines Fluidsystems muss der Benutzer ein oder mehrere TFO auf einem ATHLET-Workspace platzieren. Ein Doppelklick auf die schematische Darstellung eines TFO (das Pipe-Objekt) öffnet die Geometrieansicht. Im Beispiel (siehe Abb. 4.4) besteht das TFO (Pipe) aus einem geraden Rohrstück mit der Länge 1 m (Maßstab 100:1) und einem Durchmesser von 0,5 m mit horizontalem Verlauf (geodätische Höhe 0 m). Am Anfang und Ende des Rohrstücks sind Anschlüsse (Verbindungsunkte) definiert. Außerdem enthält das Rohrstück ein Kontrollvolumen. Die Grenze des Volumens wird durch eine dicke schwarze Linie am Ende des Rohrstücks angezeigt. Definiert wird das Kontrollvolumen durch ein CV-Boundary-Objekt (roter Pfeil).

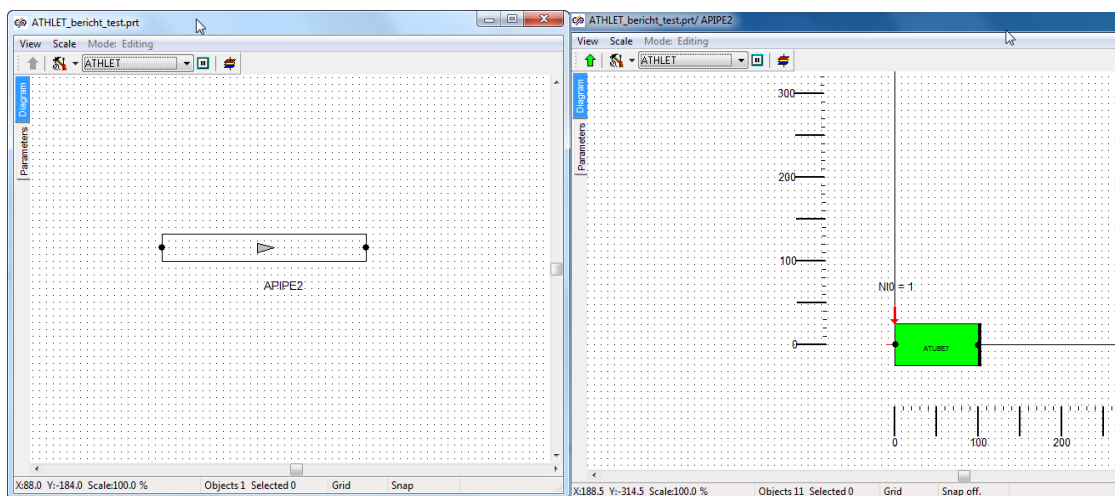


Abb. 4.4 Schematische und geometrische Darstellung eines TFO in ATM

Falls keine Vorgaben erfolgen, wird das erste Basiselement immer an den Beginn des Ursprungs gelegt und somit angenommen, dass sich die absoluten Koordinaten aus dem vorhergehenden Fluidobjekt ableiten lassen. Alternativ kann man vor das erste Basiselement einen Ortspunkt (ATH_POINT_3D) einfügen, der die absoluten Koordinaten des TFO festlegt. Diese Daten werden in der Geometriedarstellung berücksichtigt (siehe Abb. 4.5).

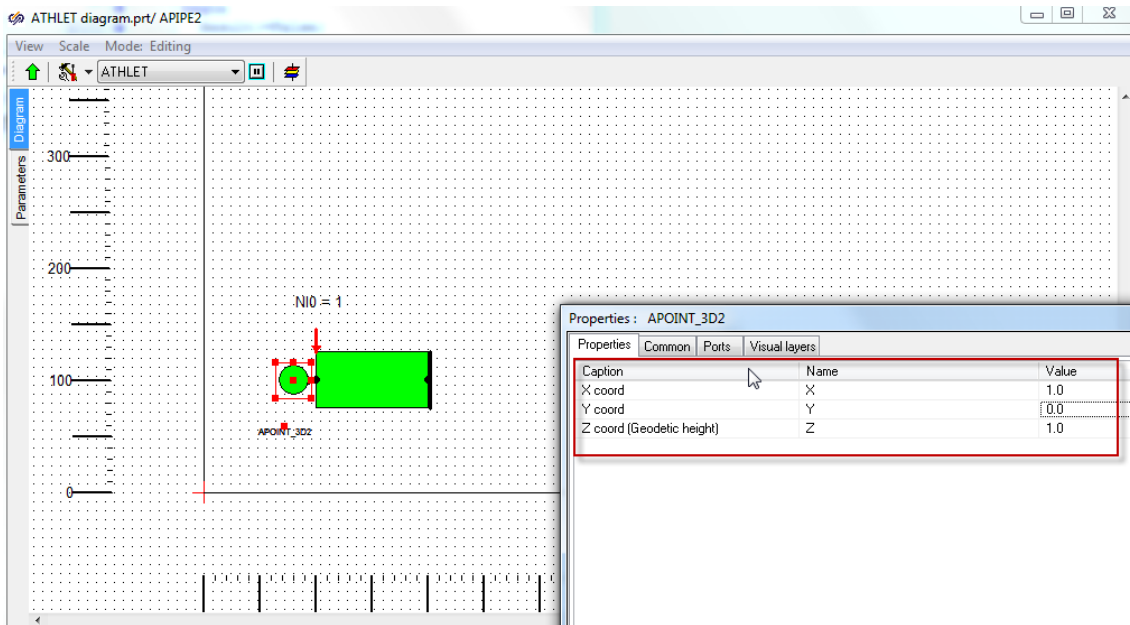


Abb. 4.5 Vorgabe der Anfangskordinaten eines TFO

Das Modell kann entsprechend den realen Gegebenheiten in der Leitung im Fluidsystem und den Modellierungsoptionen in ATHLET sukzessive erweitert werden. Dies ist durch das Hinzufügen von weiteren hydraulischen Basiselementen oder durch das Einfügen von Komponenten (Abb. 4.6) innerhalb dieser Elemente möglich.







- Interner Abzweig (zu anderen TFO) 
- Pumpe 
- Ventil 
- Doppelendiger Bruch 
- Ausströmventil 
- Ein- /Auspeisung (Fill/leak) 

Abb. 4.6 Verfügbare Komponenten für ein TFO

Das 2. Beispiel zeigt die Erweiterung des Pipes mit einem Rohrbogen. Dazu ist es zunächst nötig, das Property-Fenster (Abb. 4.7) für das Rohrstück anzuzeigen. Dieses zeigt die Daten des Rohrstücks, z. B. die Länge in X-Richtung (X length) und ob weitere Geometrieelemente folgen (Is_End). Die ursprüngliche Länge wurde auf 2 m geändert, was unmittelbar in der grafischen Darstellung (Abb. 4.8) angezeigt wird.

Für die geometrischen Basiselemente ist die Veränderung der Ausdehnung mit der Maus (Standardfunktion in ATM) deaktiviert, um die geometrischen Daten stets korrekt darzustellen. Bei Bedarf kann der Nutzer durch das Setzen der „Resize_Blocked“ Eigenschaft des Elements auf „No“ diese Funktion reaktivieren.

Caption	Name	Value
X length	XL	2.0
Y length	YL	0.0
Z length (Geodetic height)	ZL	0.0
Left diameter	DL	0.5
Right diameter	DR	0.5
Left DEPO	DEPL	0.0
Right DEPO	DEPR	0.0
Length	L	2
Left cross sectional area	AL	0.0
Right cross sectional area	AR	0.0
Volume	V	0.0
Is_Start	Is_Start	Yes
Is_End	Is_End	No
Lengths include bend	Virtual_Lengths	No
Straight Length	ST_L	2
Geodetic height straight part	ST_Z	0
Resize_Blocked	Resize_Blocked	Yes
Total length(at end)	SG0	2
Total geodetic height(at end)	Z0	0

Abb. 4.7 Eigenschaften für ein Rohrstück (Property-Fenster)

Zusätzlich wurde ein Bogenelement mit zwei Segmenten auf den Workspace gezogen und mit dem Rohrstück verbunden. Dies geschieht durch das Einfügen einer hydraulischen Verbindungslinie zwischen den Ports von Rohrstück und Bogen (Abb. 4.8). Danach wird die Position des Bogens automatisch korrigiert, um die geometrische Lage korrekt darzustellen. Ebenfalls ist die Kontrollvolumengrenze an die richtige Position (Ende des Rohrbogens, Abb. 4.9) verschoben.

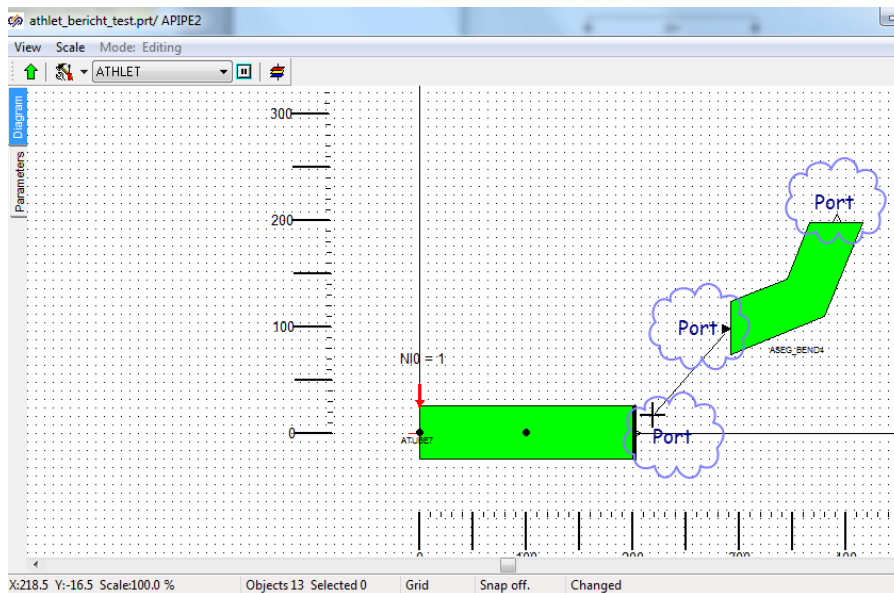


Abb. 4.8 Einfügen und Verbinden von hydraulischen Basiselementen

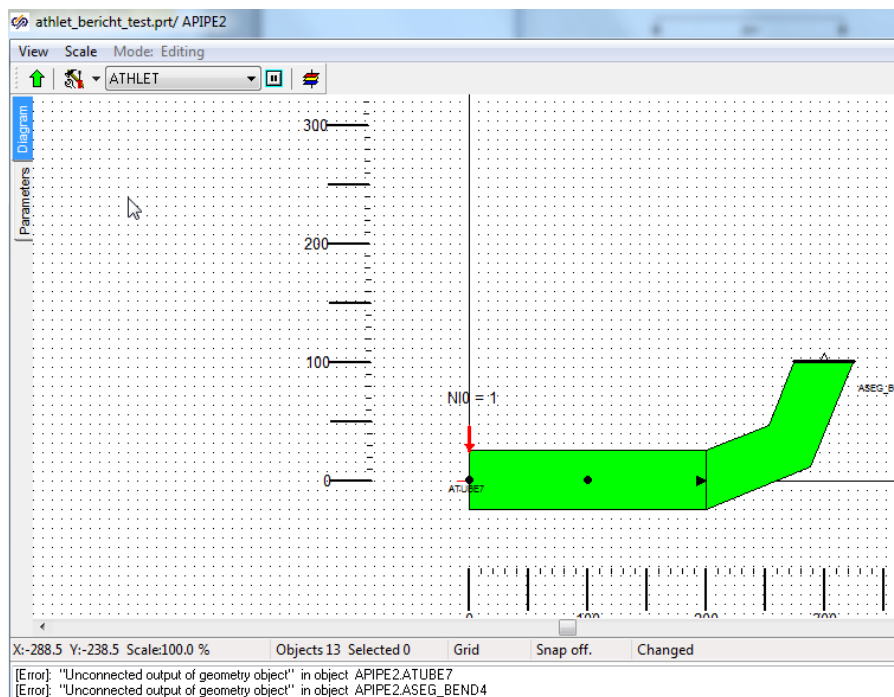


Abb. 4.9 Geometriedarstellung nach Verbindung der Basiselemente

In Beispiel 3 (Abb. 4.10) ist die geodätische Höhe (Z length, Abb. 4.7) des Rohrstücks auf 0,5 m und die Segmentzahl des Bogens auf 10 erhöht worden, was bereits zu einer sehr guten Annäherung für einen Kreisbogen führt. Gleichzeitig ist die Anzahl der Kontrollvolumen auf drei erhöht worden ($NI0=3$). Die Kontrollvolumengrenzen liegen – bezogen auf die Mittellinie – äquidistant verteilt in den geometrischen Basiselementen. Auch

die Positionen aller anderen Objekte sind automatisch entsprechend der neuen Geometrie in der Darstellung angepasst worden.

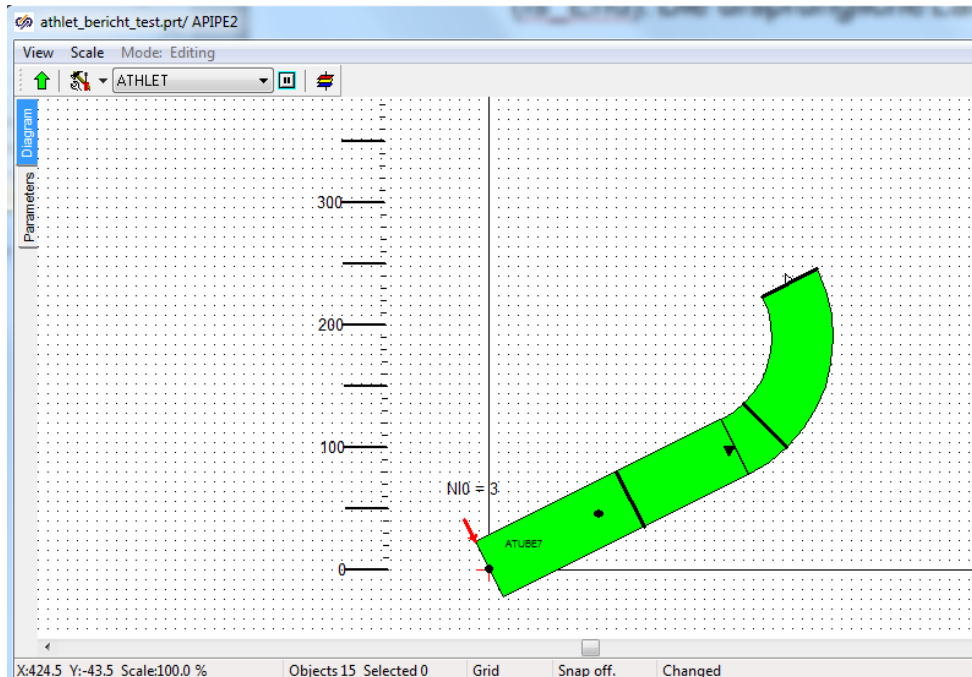


Abb. 4.10 Datenänderung in den Basiselementen des TFO

Im 4. Beispiel (Abb. 4.12) werden Komponenten, eine Pumpe und eine Einspeisung (fill) hinzugefügt und die Position eines Abzweigs (internal branch) angepasst. Grundsätzlich wird eine Junction als Teil eines Basiselements definiert, indem es auf diesem platziert wird. Anschließend muss noch die geometrische Position innerhalb dieses Elements numerisch vorgegeben werden (LPOS in Abb. 4.11). Weitere von ATHLET benötigte Daten der Junction, wie z. B. die Signale für Pumpendrehzahl und Motorschaltung (SGPUMP, SGMOT), können an dieser Stelle ebenfalls vorgegeben werden.

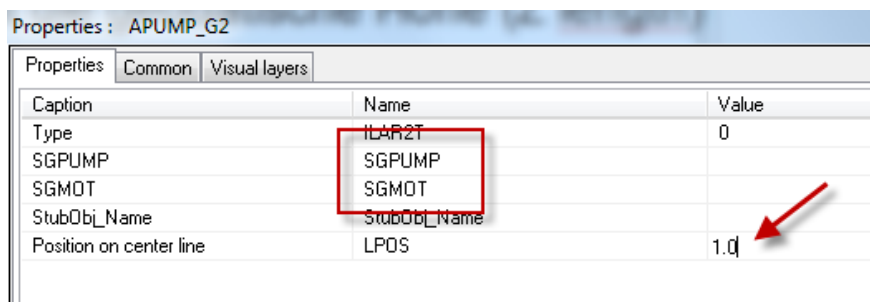


Abb. 4.11 Eigenschaften für eine Komponente (Pumpe)

Die Position der Junctions wird automatisch grafisch am korrekten Ort angezeigt. Werden die Geometriedaten der Basiselemente anschließend nochmals geändert, wird auch die Lage der Junctions in der Darstellung entsprechend angepasst.

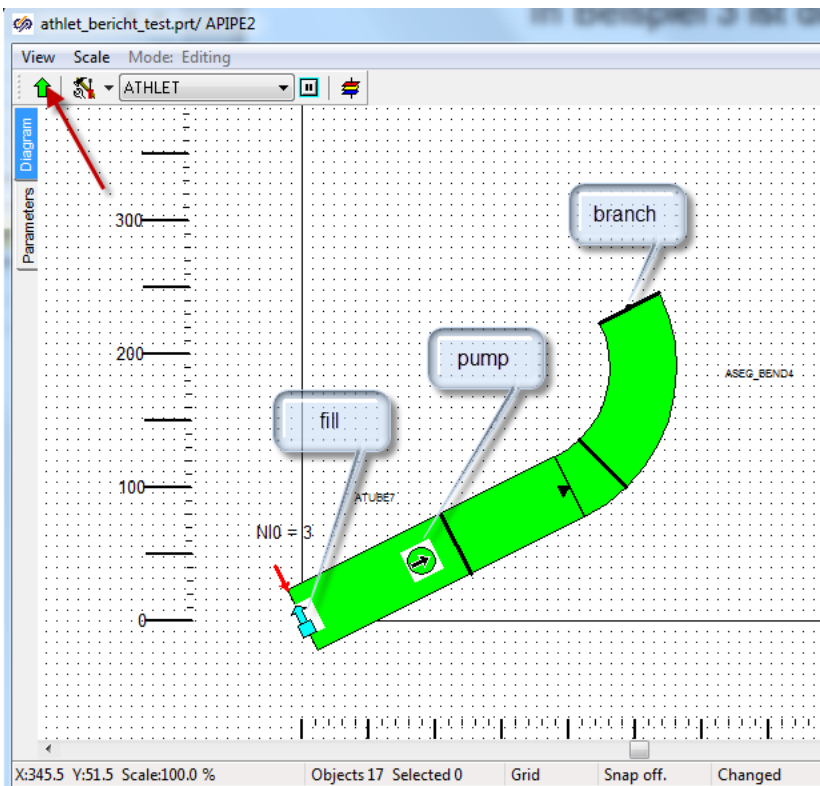


Abb. 4.12 Automatische Positionierung von Komponenten

Hat man alle notwendigen hydraulischen Elemente eines TFO vorgegeben, kann man durch einen Klick auf den grünen Pfeil in der Geometrieansicht (Abb. 4.12) wieder zum Workspace der TFO des Fluidsystems zurückkehren. Wenn man neue Junctions eingefügt hat, wird empfohlen die schematische Darstellung des Pipes zu aktualisieren. Dies ist mit dem entsprechenden Eintrag im ATHLET-Menü (Abb. 4.13) möglich. Das ATHLET-Menü ist über die Hauptmenüleiste (Abb. 4.1 rechts) von ATM erreichbar.

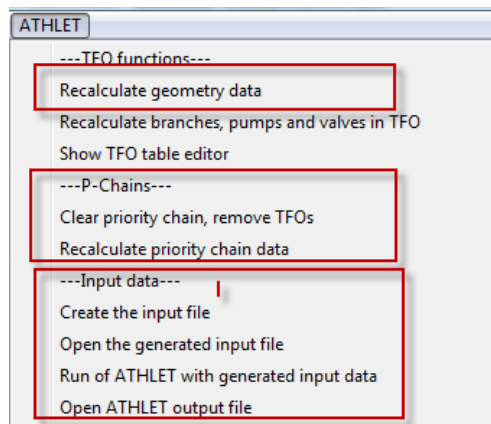


Abb. 4.13 ATHLET-spezifisches Menü in ATM

Auch die internen Abzweigungen werden dann neu erzeugt, was dazu führt, dass eventuell bereits vorhandene Verbindungen zu anderen TFO gelöscht werden und das TFO erneut im Fluidnetzwerk verbunden werden muss. Die schematische Darstellung enthält im Anschluss Symbole für die im Pipe vorhandenen Komponenten (Abb. 4.14). Die Position dieser Symbole entspricht der Position auf der abgewickelten Länge des Pipes. Eine Änderung der Position ist in dieser Ansicht nicht möglich, andere Daten der Junction können aber über das Property-Fenster geändert werden.

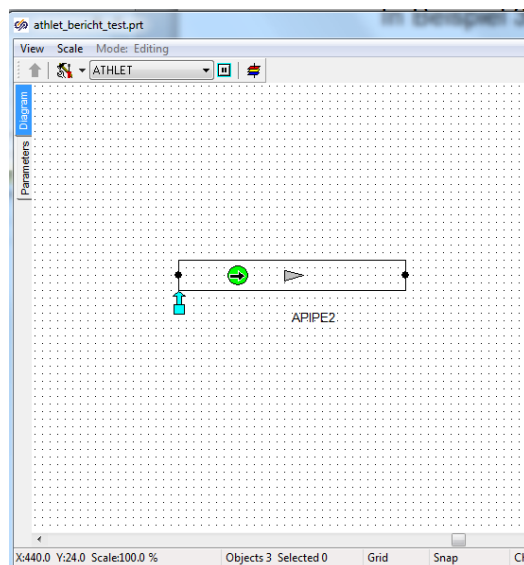


Abb. 4.14 Darstellung von Komponenten in der schematischen Ansicht

Eine numerische Anzeige der Geometriedaten eines TFO ist in der oberen Tabelle einer speziellen Dialogmaske (Abb. 4.15), der „TFO table editor“, verfügbar. Diese wird standardmäßig angezeigt, wenn die Geometrieansicht eines TFO aktiviert ist.

Sie kann aber auch jederzeit über das ATHLET-Menü mit „Show TFO table editor“ ein-
geblendet werden. Die Daten in der Tabelle werden aktualisiert, sobald in der Geometrie-
ansicht Daten geändert werden, z. B. eine Längenänderung in einem Basiselement.
Die Tabelle selbst ist nicht editierbar, dient also nur zur Anzeige der Werte, wie sie für
die ASCII-Eingabe von ATHLET generiert werden.

Object Name: AP1PE2

Automatic values (From Geometric Subobjects):

S0	N10	Z0	D0	A0	V0	DEP0	JTYP0
0		0	0.5	0	0	0	
1							2
2.2361		1	0.5	0	0	0	
2.3931		1.081	0.5	0	0	0	
2.5502		1.182	0.5	0	0	0	

Branching / BranchZM:

Other local values (Manually supplied):

S1	SDFJ0	ZFFJ0	ZFBJ0	JFLO0	JDRIFT	P0	T0	G0	Q0	ICK00
0				2	1	1.E5	30.0	1	0	0
1						0	0	1	0	0

Buttons: Insert Row, Delete Row, Copy Row, Paste Row, Apply Values, Restore Values

Abb. 4.15 Anzeige von ortsabhängigen Daten im TFO table editor

Weitere ortsabhängige Daten für das TFO, wie z. B. lokale Anfangswerte für Druck und
Temperatur, Reibungsbeiwerte u. a. können in der unteren Tabelle für beliebige Längs-
koordinaten vorgegeben werden, indem mit „Insert Row“ weitere Zeilen zur Tabelle hin-
zugefügt werden. Die Werte einer Zeile können mit „Copy/Paste Row“ in eine andere
Zeile übernommen werden. Werte aus Zellen der oberen Tabelle können mittels Klick in
die jeweilige Zelle in die Zwischenablage kopiert und auf diese Weise bei Bedarf einge-
fügt werden. Alle Tabellenwerte sind als Properties eines TFO gespeichert. Erst mit Be-
tätigen des Buttons „Apply Values“ werden die Tabellenwerte in die Properties des TFO
übertragen. Die aktuell in den Properties vorhandenen Werte kann man mit „Restore
Values“ wiederherstellen. Eine permanente Speicherung der Werte erfolgt aber erst mit
dem Speichern des Fluidsystems im *.prt-File.

Das Branch-Objekt, das in ATHLET zur Modellierung von Verzweigungen, wie z. B. im
oberen Plenum des RDB, verwendet wird, benötigt ähnliche Daten wie das Pipe-Objekt.
An einem Branch-Objekt können beliebig viele Leitungen (Pipe-Objekte) angeschlossen
sein.

Die Anzahl und Lage der Anschlüsse kann in den Properties (Number of junctions) eingegeben werden. Der TFO table editor stellt für diese Branches eine weitere Tabelle (Abb. 4.16) zur Verfügung, die die Vorgabe von „Branching“ und „Branch2M“ Daten für ATHLET ermöglicht entsprechend der Eingabedatenbeschreibung in /LER 16/. Dort können die Daten zum Impulsaustausch von Pipes, die über das Branch verbunden sind, definiert werden.

Branching / Branch2M:

Pipe	IDPR	ADPR	Coupl Pipe
P1-HL	1	1.0	PV-COR.1
			PV-COR.1
			P1-HL

Abb. 4.16 Vorgabe von „Branching“ und „Branch2M“ Daten im TFO Table Editor

4.1.2.2 Implementierung der Geometriedarstellung eines TFO

Es hat sich gezeigt, dass eine korrekte automatisierte Darstellung der Geometrie beim Verbinden oder Ändern einzelner Basiselemente einen erheblichen, nicht vorhergesehenen Programmieraufwand erforderte.

Zunächst werden wesentliche Punkte genannt, die für eine korrekte, weitgehend automatisierte Darstellung der geometrischen Daten mit der Diskretisierung in Nodes notwendig sind:

- Änderung von Länge, Breite und Winkel und Position der Geometrie-Elemente
- Berechnung des einhüllenden Rechtecks (Echo Area) der Geometrie-Elemente
- Berücksichtigung eines Anfangswinkels bei Bögen (für einen stetigen Anschluss)
- Berücksichtigung eines Bogenwinkels zwischen 0 und 180 Grad
- Berücksichtigung einer Bogenrichtung (Biegung nach oben oder unten)
- Berücksichtigung einer beliebigen Anzahl von diskreten Segmenten bei Bögen
- Berechnung der Position der Verbindungsports

- Berechnung der Position der Objekte entsprechend der korrekten Längs- und Höhenkoordinaten oder des vorhergehenden Objekts
- Berechnung von Position, Länge und Winkel der Nodegrenzen
- Berechnung der Lage der Junctions

Im Basissystem SimInTech werden grafische Objekte mit Punktkoordinaten in der x,y-Ebene für das einhüllende Rechteck beschrieben. Zusätzlich ist ein Drehwinkel um das Objektzentrum vorhanden. Für die Punkte der Vektorgrafik des Objekts werden die Koordinaten relativ zum Objektzentrum angegeben, ohne Berücksichtigung des Drehwinkels. Die Vektorgrafik muss die Objektgeometrie maßstabsgetreu wiedergeben.

Alle Punktkoordinaten müssen erneut für alle oder einige Geometrieobjekte berechnet werden, sobald sich die geometrischen Daten von Vorläuferobjekten oder die Nodalisierung vom Benutzer geändert oder neue Geometrieobjekte in die Kette eingefügt werden. Dazu wurden geeignete Formeln unter Berücksichtigung der Trigonometrie und der nötigen Transformationen entwickelt. Diese sind dann im Programm in Funktionsroutinen implementiert worden, die bei einer Datenänderung der Geometrieobjekte automatisch ausgeführt werden. Besonders komplex sind die Formeln für die Neuberechnung der Rohrbogenelemente, da die Punktkoordinaten z. B. von der Segmentzahl und dem Bogenwinkel abhängen.

Das Verhalten der Geometriedarstellung wurde an einer Vielzahl von Beispielen getestet und Fehler in der Darstellung sukzessive beseitigt. Die erreichte Qualität der Geometrieansicht (Abb. 4.17) eines TFO ergibt ein sehr gutes Abbild der modellierten Leitungsgeometrie und hilft, Eingabefehler frühzeitig zu erkennen. Alle Geometrie-Elemente werden bezüglich Ausdehnung und Lage entsprechend ihrer Daten maßstabsgetreu dargestellt. Mit dem Konzept für die Nodalisierungsvorgabe durch die CV-Boundary-Objekte (rote Pfeile in Abb. 4.17), mit Ort und Anzahl der Nodes, wird die Geometriedarstellung entsprechend geändert und die Lage aller aktuell definierten Nodegrenzen grafisch mittels Trennlinien angezeigt. Der Benutzer erhält damit unmittelbar bei der Eingabeerstellung einen Eindruck, ob an wichtigen Stellen des TFO die Diskretisierung hinreichend genau vorgegeben ist. Dies war bisher nur nach der Durchführung einer Simulationsrechnung mit Hilfe der ATHLET Input Grafik möglich.

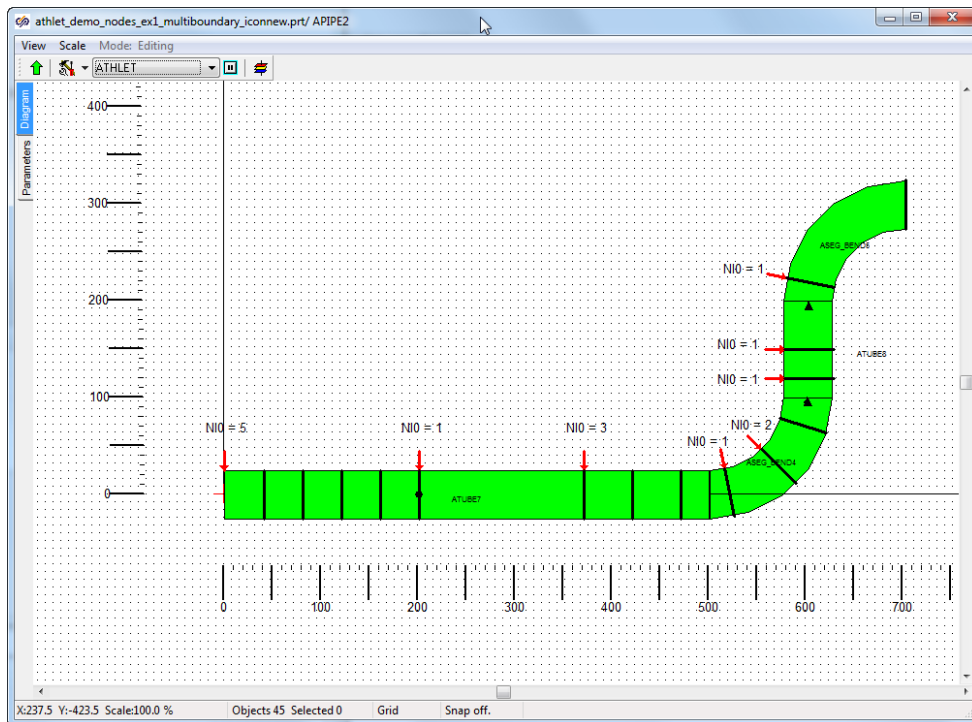


Abb. 4.17 Beispiel für ein TFO mit detaillierter Diskretisierung

4.1.2.3 Formverlustkoeffizienten

Für einige hydraulische Standardkomponenten gibt es empirische Näherungen zur Ermittlung der Reibungsdruckverluste. Diese müssen als „form loss coefficients“ für TFO im ATHLET-Eingabedatensatz vorgegeben werden. ATM kann bei der Berechnung dieser Formverlustkoeffizienten (Widerstandzahl, Zetawert) von einigen hydraulischen Rohrleitungselementen unterstützen.

Gegenwärtig sind Algorithmen für Rohrbogen, Kniestück (Knick), Diffusor und Düse implementiert. Die Berechnungen erfolgen auf Basis von empirischen Grundlagen aus Idel'Chik /IDE 66/ oder dem VDI-Wärmeatlas /VDI 84/. Für Idel'Chik wurden die Zeta-werte entsprechend der Kurven mit Funktionen approximiert, die Werte des VDI Wärmeatlas sind tabellarisch vorgegeben und werden interpoliert. Alle Werte werden für glatte Rohrstücke mit hohen Reynoldszahlen ermittelt. Die verwendeten Methoden zur Berechnung der Formverluste sind detailliert im Anhang A beschrieben.

Im Property Menü eines Diffusors (Abb. 4.18) kann man die Berechnungsmethode über ein Pulldown-Menü auswählen.

Beim Speichern der Auswahl mit „OK“ werden die Formverlustbeiwerte für Vorwärts- und Rückströmung durch den Diffusor mit der nötigen Normierung berechnet und angezeigt. In der Abbildung sind zusätzlich die verschiedenen Werte aus den beiden Berechnungsmethoden eingeblendet.

Properties	Common	Ports	Visual layers
Caption	Name	Value	
X length	XL	1.202	
Y length	YL	0.0	
Z length (Geodetic height)	ZL	0.0	
Left diameter	DL	0.365	
Right diameter	DR	0.63	
Form loss calc. method	F_Method	VDI Atlas	
Zeta	Zeta	Manual	
Form loss coefficient (fw)	ZFFJO	Idelchik	
Form loss coefficient (bw)	ZFBJO	VDI Atlas	
Form loss calc. method	F_Method	VDI Atlas	
Zeta	Zeta	0.088130003	
Form loss coefficient (fw)	ZFFJO	8.0495678	
Form loss coefficient (bw)	ZFBJO	3.6534971	
Form loss calc. method	F_Method	Idelchik	
Zeta	Zeta	0.078821848	
Form loss coefficient (fw)	ZFFJO	7.1993849	
Form loss coefficient (bw)	ZFBJO	5.4636211	

Abb. 4.18 Formverlustkoeffizienten für einen Diffusor mit verschiedenen Auswahlmöglichkeiten

Die Implementierung der Funktionen zur Berechnung für alle unterstützten Elemente und jeweils verschiedenen Geometrien wurde einer Vielzahl von Tests unterzogen. Die Ergebnisse beider Verfahren wurden mit der Literatur und untereinander verglichen. Die gefundenen Abweichungen waren entsprechend der verwendeten Näherungen plausibel und tolerierbar. Für einen speziellen Fall lagen dokumentierte Anlagedaten vor und die per ATM errechneten Formverluste zeigten auch hier eine gute Übereinstimmung.

Im Ergebnis vereinfacht die automatisierte Berechnung der Formverlustbeiwerte eine korrekte Vorgabe, insbesondere für wenig geübte Anwender, und hilft, Eingabefehler zu vermeiden.

4.1.2.4 Prioritätsketten

In ATHLET werden die Verbindungen der Fluidobjekte (TFO) in sogenannten Prioritätsketten definiert. Diese geben an, welche Objekte miteinander verbunden sind und definieren zusätzlich die Reihenfolge der Berechnung für die stationäre Lösung. Sie geben die Position der Verbindung (Ortskoordinaten) zwischen den TFO an. Jedes TFO muss

zumindest in einer Kette enthalten sein. In ATHLET sind zur Bereitstellung dieser Information „Priority Chain“-Objekte (PCO) implementiert worden.

Die Vorgehensweise zur Modellierung eines Fluidsystems mit zwei PCO wird im Folgenden an einem einfachen Beispiel erläutert, das Teil eines Musterdatensatzes von ATHLET ist. In Abb. 4.17 ist ein einfaches thermohydraulisches Modell für einen Kühlkreislauf in einem Druckwasserreaktor dargestellt, welches zwei Branches und sieben Pipes verwendet. Die Verwendung mehrerer TFO Leitungs- und Knotenobjekte in einem Diagramm und ihre Verbindung mit Strömungslinien (Hydraulical Wires) erlaubt die Modellierung des Kreislaufs mit diesen Objekten. Die Richtung der Hydraulical Wires kann beim Verbinden von TFO beliebig gewählt werden. Sie muss nicht notwendigerweise der stationären Strömungsrichtung entsprechen. Die Lage, Länge und Ausrichtung der einzelnen TFO in der Übersichtsdarstellung ist manuell so angepasst, dass bereits ein Eindruck der Geometrie des gesamten Kreislaufs entsteht. Die Anpassung ist ohne Einfluss auf die erzeugten ATHLET Daten.

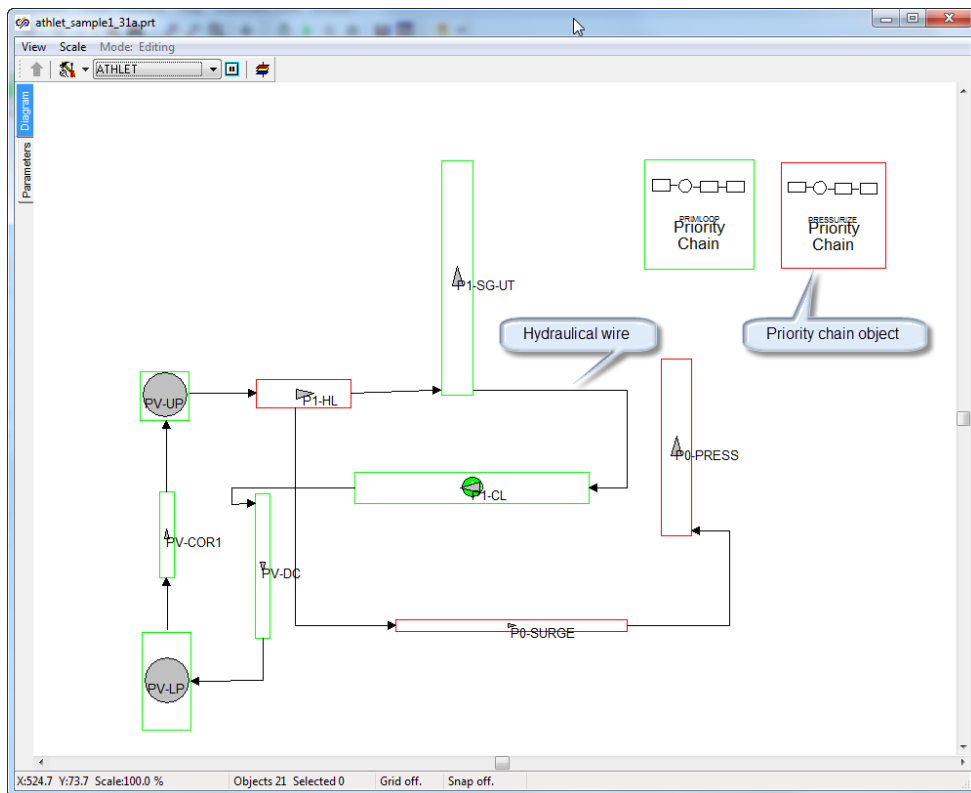


Abb. 4.19 Modellierung eines Fluidkreislaufs in ATHLET

Die einzelnen TFO werden, wie in Abschnitt 4.1.2.1 beschrieben, mit geometrischen Bauelementen zusammengesetzt und hinsichtlich der hydraulischen Komponenten exakt modelliert und dargestellt.

In Abb. 4.20 ist das Modell einer Leitung (P0-SURGE) zu sehen, welches die Volumenausgleichsleitung von der heißen Kühlmittelleitung (P1-HL) zum Druckhalter (P0-PRESS) modelliert. Bereits an dieser Stelle werden die Positionen der Verbindungen zu anderen TFO festgelegt, die dann in den PCO verwendet werden.

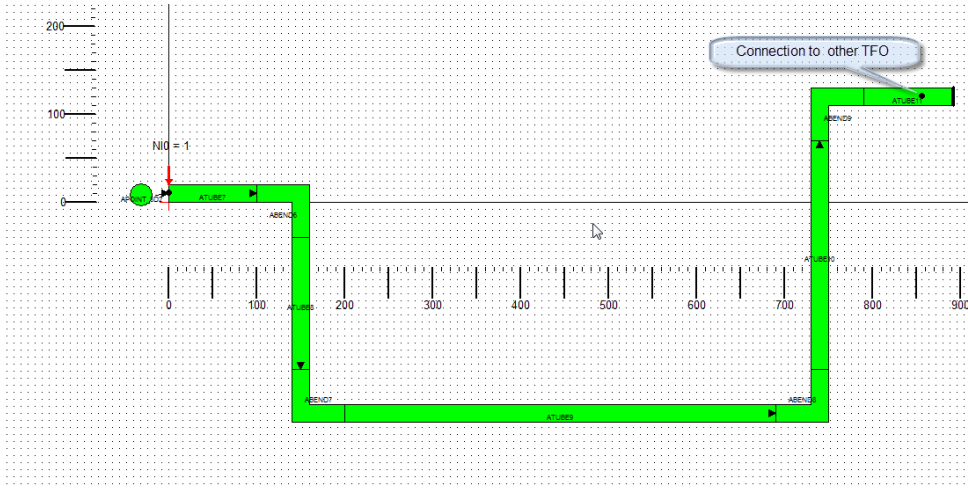


Abb. 4.20 Modell einer Leitung (Surgeline) im Fluidsystem

Für jede benötigte Prioritätskette ist ein eigenes PCO nötig. Im Beispiel werden zwei PCO verwendet, mit den Namen PRIMLOOP und PRESSURIZE. Die Zuordnung von Fluidobjekten zu einer PCO ist durch die Selektion der gewünschten Objekte und das Kontextmenü der PCO möglich (Abb. 4.21), wobei mehrere TFO gleichzeitig ausgewählt und in das PCO eingefügt werden können. Bei der Bestimmung der Objekt-Reihenfolge wird in ATM die Richtung der hydraulischen Verbindung verwendet. Die Berechnungsreihenfolge wird somit durch die Richtung der Hydraulical Wires festgelegt. TFO können zu einer bestehenden Kette nur hinzugefügt werden, wenn das vorangehende Objekt bereits in der Kette enthalten ist. Ein Beginn einer neuen Kette ist hingegen in einem beliebigen TFO jederzeit möglich. Die Zugehörigkeit eines TFOs zu einer Prioritätskette ist durch die verwendete Konturfarbe erkennbar.

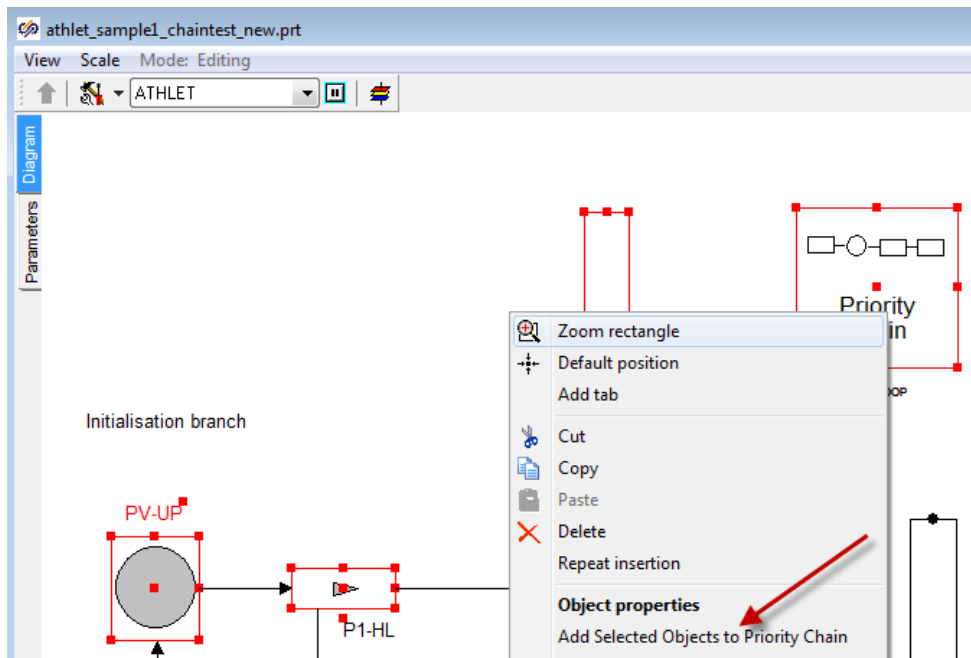


Abb. 4.21 Zuordnung von TFO zu Prioritätsketten

Wenn am Fluidnetzwerk Änderungen vorgenommen werden müssen die Prioritätsketten neu berechnet werden. Dazu gibt es im ATHLET Menü die in Abb. 4.13 gekennzeichneten Optionen. Bei Änderungen der Struktur des Fluidnetzwerks, z. B. durch Einfügen oder Löschen eines TFO, müssen alle Daten der PCO, die das TFO enthalten, zunächst gelöscht und anschließend wieder alle gewünschten TFO zu den PCO hinzugefügt werden. Bei Änderungen der Ortskoordinaten der Verbindungspunkte in den Geometriedaten der TFO müssen diese Änderungen in die Neuberechnung der PCO-Daten übernommen werden.

Gegenüber der bisher üblichen Eingabeerstellung für Prioritätsketten ergibt sich eine deutliche Vereinfachung für den Nutzer. Es ist kaum eine alphanumerische Eingabe mehr notwendig, da sich Verbindungen und Verbindungsorte automatisch aus dem erstellten Fluidsystem in ATM ableiten lassen. Lediglich der Name der PCO muss angegeben und die in der Prioritätskette vorhandenen Objekte müssen interaktiv ausgewählt und hinzugefügt werden.

4.1.2.5 Modellierung eines HCO

Wärmeleitobjekte (heat conduction objects, HCO) dienen in ATHLET beispielsweise zur Nachbildung von Wärmeverlusten über Strukturen, für Wärmetauscher oder Brennstäbe. Sie sind häufig mit einem einzelnen TFO verbunden, welcher die Temperaturverteilung am Rand des HCO vorgibt.

In einigen Fällen kann ein HCO gleichzeitig mit zwei TFOs verbunden sein kann, wie z. B. für die Modellierung eines Wärmetauschers mit TFOs für Primär- und Sekundärseite. Dafür musste das Basissystem SimInTech von ATM so erweitert werden, dass verschiedene grafische Objekte auf dasselbe HCO und die entsprechenden Objektdaten verweisen können.

Im Folgenden wird ein Beispiel für die Modellierung eines Wärmeleitobjekts gezeigt. In Abb. 4.22 ist die schematische Darstellung eines Wärmeleitobjekts HCO1 sichtbar, das mit zwei Fluidobjekten, TFO1 und TFO2, verbunden ist und somit eine Wärmekopplung zwischen diesen TFOs simuliert.

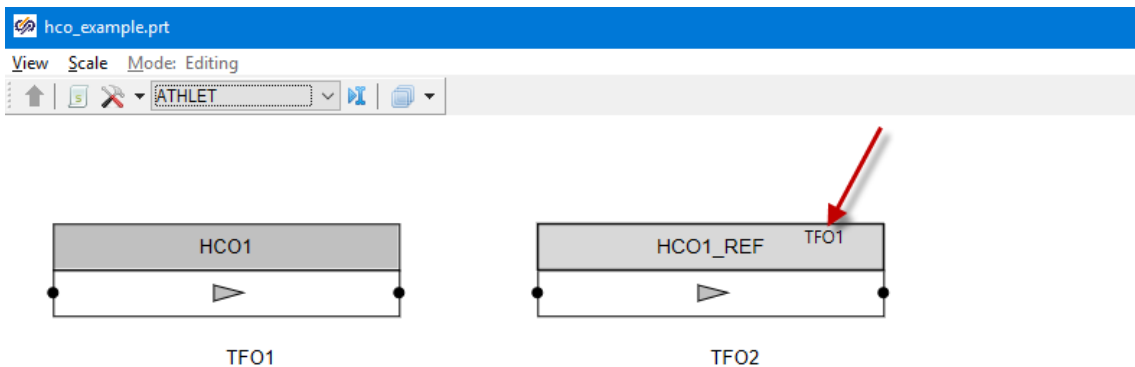


Abb. 4.22 Kopplung zweier TFOs mit einem HCO

Ein Wärmeleitobjekt kann in ATHLET mit einer Vielzahl von Daten beschrieben werden, die teilweise optional sind. Diese Daten können im Property Dialog (Abb. 4.23) des HCO vorgegeben werden. Optionale Daten können über diesen Dialog ein- und ausgeschaltet werden.

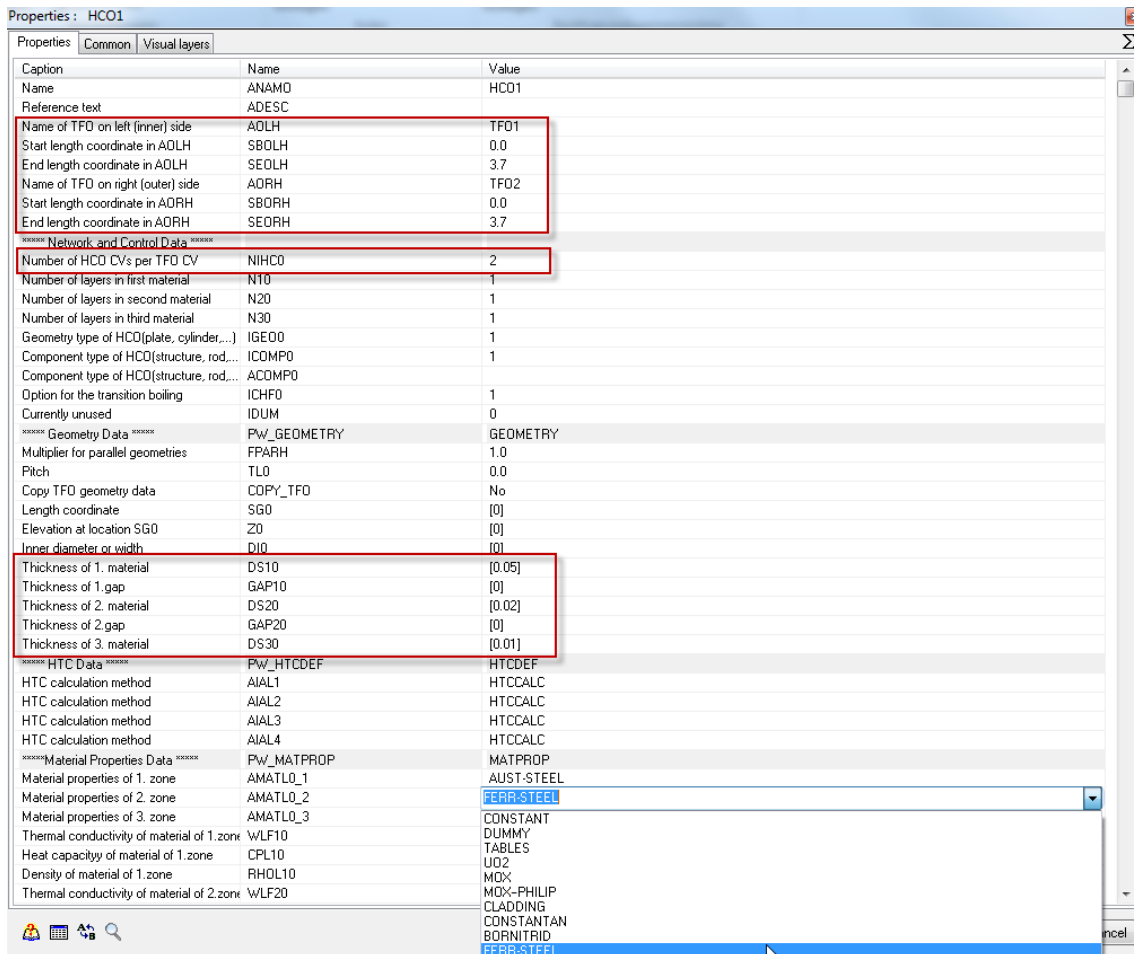


Abb. 4.23 Property Dialog für ein Wärmeleitobjekt

In der Abbildung sind wichtige Daten, wie die gekoppelten TFO und die Materialdicken hervorgehoben. Die Materialtypen der Schichten lassen sich über ein Dropdownmenü auswählen. Bereits beim Erzeugen eines HCO über das ATHLET Menüleiste mit **HCO** wird per Drag-and-Drop die Zuordnung zu einem TFO zur linken Seite des HCO festgelegt. Bei Bedarf kann auf ähnliche Weise, unter Nutzung einer HCO-Referenz **HREF** auch die rechte Seite des HCO einem TFO zugeordnet werden. Im HCO-Referenzobjekt HCO1_REF wird der Name des TFO auf der linken Seite eingeblendet (Pfeil in Abb. 4.22). Mit einem Doppelklick auf das Referenzobjekt HCO1_REF wird das referenzierte HCO1 automatisch selektiert.

4.1.2.6 Modellierung eines CCO

Die vollständige Bezeichnung eines Cross Connection Objects (CCO), wie sie im ATHLET User Manual steht, lautet "Cross Connection Object for parallel Channels". Diese vollständige Bezeichnung erklärt den Zweck dieses Elementes – Verbindung von

parallelen Kanälen, in deren Rolle normalerweise Pipes oder Branches auftreten. Technisch wird in ATHLET für ein CCO eine Serie von Junctions (Verbindungen) zwischen entsprechenden CVs (Control Volumes) von den beiden verbundenen Kanälen automatisch generiert. Die Eigenschaften von CCO geben an, wie genau der Austausch zwischen den beiden Kanälen, die keine direkte Verbindung miteinander aufweisen, abläuft.

ATM stellt zur Modellierung eines CCO über ATHLET Menüleiste einen entsprechenden grafischen Block zur Verfügung, wie er in der Abb. 4.24 dargestellt ist. Das Objekt CCO1 verbindet hier zwei Pipes, Pipe01 und Pipe02, und behält diese Verbindungen auch wenn die entsprechenden Pipe-Objekte umbenannt werden.

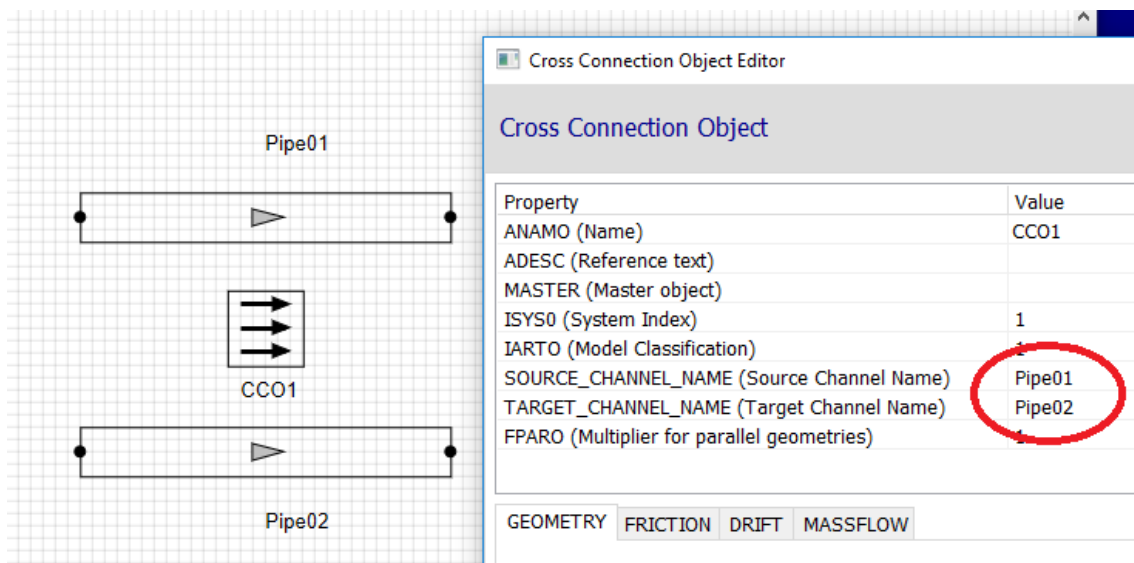


Abb. 4.24 Ein CCO verbindet zwei parallele Pipes

In der ATHLET-Terminologie stellt ein Cross Connection Object ein spezielles TFO dar und weist damit ähnliche (aber nicht genau gleiche) Eigenschaften wie ein TFO auf, die in einem speziellen Editor verwaltet werden können, wie in der Abb. 4.24 ebenfalls zu sehen ist.

Für die derzeitige Entwicklung von ATM stehen aktuell 2 Editoren für ein CCO-Block zur Verfügung, wobei diese Möglichkeiten insbesondere auch zur weiteren Erprobung von ATM dienen. Ein CCO-Block kann in einem im Basissystem SimInTech standardmäßig vorhandenen generellen Editor bearbeitet werden. In Abb. 4.25 stehen dieser standardmäßige Editor (links) und ein neuer Editor (rechts) zum Vergleich nebeneinander.

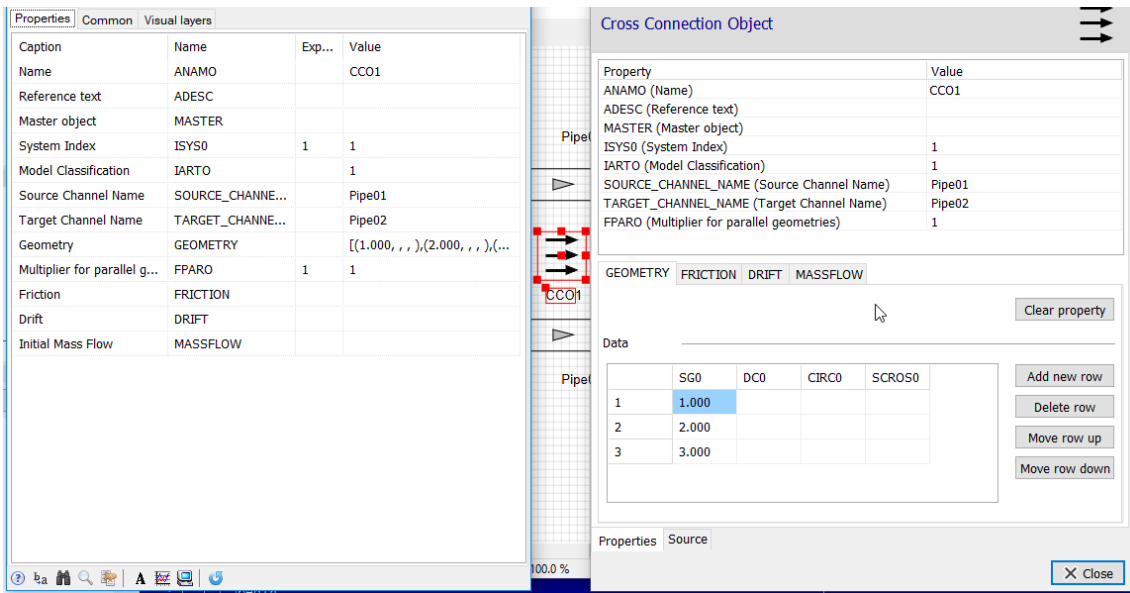


Abb. 4.25 Alte (links) und neue (rechts) Editoren für ein CCO

Der neue Editor bietet Bearbeitungsmöglichkeiten, die besser an die Datentypen der konkreten Eigenschaften angepasst sind, wie, zum Beispiel, tabellarische Darstellungen für mehrdimensionale Daten. Des Weiteren können mehrere Eigenschaften strukturell miteinander verbunden werden, wie DRIFT, FRICITION und MASSFLOW, die alle mit der GEOMETRIE zusammenhängen. Der neue Editor sorgt automatisch dafür, dass diese strukturelle Verbindung nicht verletzt wird, was unter Anwendung von dem Standardeditor dagegen nicht gewährleistet wird und somit eine Fehlerquelle offenlässt.

Weiterhin bietet der neue Editor auch die Möglichkeit aus den soweit angegebenen Eigenschaften des Objektes unmittelbar den Code zu sehen, der basierend auf diesen Eigenschaften für ATHLET-Input generiert wird, wie in der Abb. 4.26 dargestellt ist.

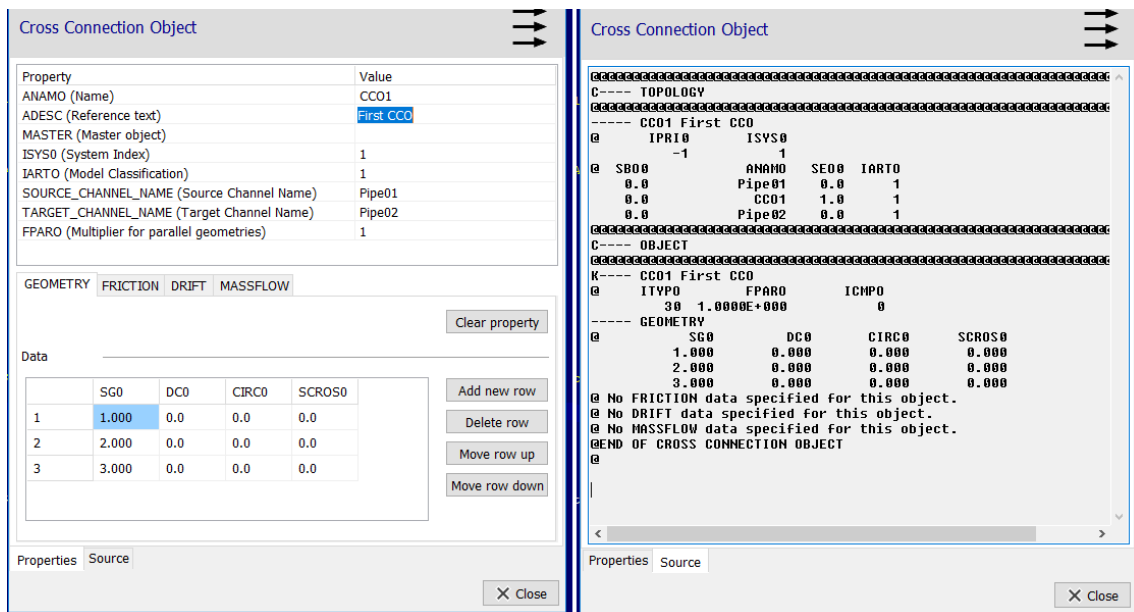


Abb. 4.26 Ad-hoc Generierung des ATHLET Codes für ein CCO

In einigen (Standard-)Fällen lässt sich diese Transformation auch umkehren – aus dem eingegebenen oder aus der Zwischenablage eingefügten ATHLET Code werden die Eigenschaften des Blocks befüllt, was als Basis für den Import von bestehenden ATHLET Datensätzen genutzt wird.

Der neue Editor ist grundsätzlich nicht nur für Cross Connection Objects konzipiert, sondern soll eine universelle sowie auch erweiterbare Basis für die Bearbeitung von allen in ATM zur Verfügung gestellten graphischen Blöcken bieten. Künftig soll dieser neue Editor mit seinen Möglichkeiten auch von weiteren Blöcken verwendet werden können.

4.1.2.7 Erzeugung und Test eines Eingabedatensatzes


Aus den interaktiv erstellten Objekten und Daten können die äquivalenten ASCII-Daten in dem von ATHLET benötigten Format erzeugt werden. ATM unterstützt die aktuelle Release-Version 3.1A. Bei einer Änderung der Eingabedaten, z. B. erweiterte Objektdaten für TFO, muss ATM entsprechend angepasst werden.

Auch wenn ATM bisher nur einen Teil der ATHLET-Eingabe-Optionen abdeckt, ist es bereits in dieser Entwicklungsphase nötig, die erzeugten Eingabedaten auf Korrektheit zu testen. Daher ist die Erzeugung der Eingabedaten aufgeteilt in eine Prozedur, die alle interaktiv definierten Objekte abarbeitet und die Verwendung von frei definierbaren

Datenzeilen für den Kopf, Beginn und Ende der gewünschten ATHLET Eingabedatei ermöglicht.

Die Prozedur prüft, ob alle Verbindungen („Ports“) der Objekte belegt und die notwendigen Objektdaten eingetragen sind. Falls keine Fehler vorhanden sind, werden die entsprechenden Zeilen für die ASCII-Eingabe erzeugt. Eventuelle Fehler, wie unverbundene Ports, werden erkannt und in der Statuszeile des Programmfensters gemeldet.

Die frei definierbaren Zeilen enthalten alle Daten, die bisher nicht interaktiv vorgegeben werden können. Mit diesem Vorgehen kann ein vollständiger ASCII-Datensatz für ATHLET vorgegeben werden. Mit der sukzessiven Erweiterung von ATM um neue Objekte, können die zusätzlichen Datenzeilen entsprechend reduziert werden. In jedem Schritt kann damit sichergestellt werden, dass die Erweiterungen zum inhaltlich gleichen Datensatz führen.

Das Starten des ATHLET Programms und auch die zuvor nötige Erzeugung der Eingabedatei sind über das ATHLET Menü möglich (Abb. 4.13). Das ausführbare Programm (Executable) und weitere Startparameter werden in den „Simulation Properties“ des Diagramms (Abb. 4.27) angegeben, die mit dem Icon  angezeigt werden können. Dort werden auch der Name der generierten Eingabedatei und die zusätzlichen Datenzeilen, mit einem Standardtexteditor, vorgegeben.

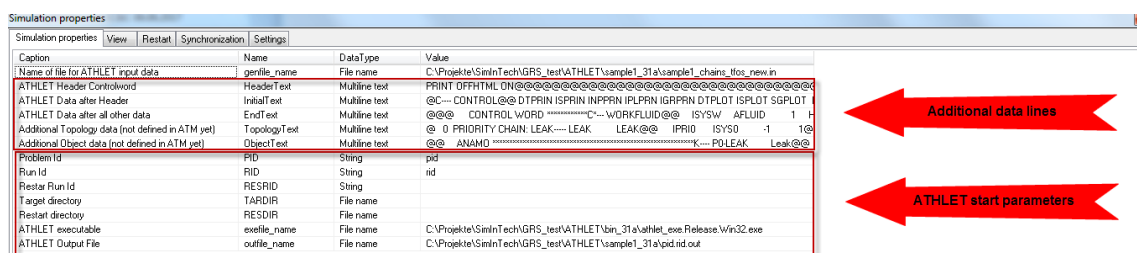


Abb. 4.27 Datenspezifikation für Eingabedatei und Simulation

Zur Kontrolle können bei Bedarf sowohl die generierte Eingabe- als auch die Ausgabedatei der ATHLET Simulation direkt in ATM im Standardeditor (siehe Abb. 4.28) angezeigt werden.

```

1 <html>
2 <head>
3 <title>Inhalt</title>
4 </head>
5 <body>
6 <pre>
7 An older .out has been overwritten, because it had the same name as the current calculation c
8
9 *****
10
11 ATHLET RUN ENVIRONMENT:
12 -----
13 Computer      : Intel64 Family 6 Model 60 Stepping 3, GenuineIntel
14 Input data    : C:\Projekte\SimInTech\GRS_test\ATHLET\sample1_31a\sample1_chains_tfos_n
15 Target Directory : .
16 Problem Id    : pid
17 Run Id       : rid
18 Executable    : UNKNOWN
19
20 *****
21
22 ATHLET PROGRAM VERSION:
23 -----
24 ATHLET 3.1A Patch#02
25 PROGRAM VERSION DATE: 16.11.16
26 PROGRAM BUILD DATE:  Nov 16 2016 16:20:33
27 PROGRAM EXTENSIONS:  None
28
29 SERIAL VERSION OF ATHLET

```

Abb. 4.28 ATHLET Ausgabe im Standardeditor von ATM

Die Erstellung eines Thermohydrauliknetzwerks wurde an verschiedenen Beispielen getestet. Das in Abb. 4.19 gezeigte Fluidsystem, ein Primärkreislaufmodell eines DWR, ist ein Teil eines ATHLET-Beispieldatensatzes. Bei Vorgabe identischer Daten liefert der von ATM erzeugte Datensatz dieselben Ergebnisse wie der Beispieldatensatz.

4.1.3 Datenimport für ATM

Für ATHLET sind eine Vielzahl von Eingabedaten in ASCII-Form vorhanden, die in einem klassischen Text-Editor erstellt wurden. Die Übertragung dieser Daten in das interaktive Eingabesystem ATM auf manuelle Art ist fehleranfällig und zeitaufwendig. Daher wurden Funktionen entwickelt, die den Import dieser Daten unterstützen.

Das Format der ATHLET Eingabedaten ist zwar strukturiert, setzt aber, wegen der historischen Entwicklung, auf keinen Standards wie z. B. XML auf. Damit ist es nicht mit Standardbibliotheken lesbar und nur schwer erweiterbar. Für alle Objekte und Modelle von ATHLET liegen spezifische Eingabeformate vor, die sich zudem, je nach gewählter Modelloption, unterscheiden können und auch noch von der ATHLET-Version abhängen. Eine genaue Beschreibung dieser Daten findet sich in der „ATHLET Input Data Description“ /LER 16/. Die genauen Details zu den Daten und Formaten sind im Eingabeparser von ATHLET programmiert. Dieser ist aber als elementarer Bestandteil von ATHLET in FORTRAN codiert und daher in anderen Anwendungen nicht verwendbar. Daher müsste der Eingabeparser in ATM in großen Teilen neu geschrieben werden, um

die Daten korrekt zu interpretieren und übernehmen zu können. Bei Änderungen der Datenformate wären dann entsprechende Anpassungen in beiden Parsern nötig. Der Aufwand bei diesem Konzept wäre unverhältnismäßig hoch. Daher wurde zunächst auf die komplette Abdeckung der ATHLET-Modelle beim Import verzichtet. Künftig sind Arbeiten vorgesehen, die auf eine Standardisierung und Flexibilisierung des Eingabeformates hinzielen. Darauf aufbauend können dann auch die Importfunktionen erweitert werden. Die bisherigen Arbeiten zum Datenimport auf Basis des existierenden Datenformats konzentrierten sich daher auf die Thermohydraulikobjekte (TFO), die ein großen Anteil zu den nötigen Eingabedaten beitragen. bereitzustellen. Dies verbessert die Akzeptanz der Anwender, ATM auch für bestehende Datensätze einzusetzen.

4.1.3.1 Datenimport für eine Eingabedatei

Die erste implementierte Funktion importiert TFOs, also Pipes, Branches und Cross-Connection-Objects, aus einer Eingabedatei für ATHLET. Dazu werden:

- die Objekte im ASCII Datensatz gesucht,
- die Objektdaten gelesen,
- die Objekte in ATM generiert,
- die Daten in die Objekt-Properties geschrieben.

```

C:\Simulators\testdata_athlet_31a\sample1.in
0 1,0 2,0 3,0 4,0 5,0 6,0 7,0 8,0 9,0
101 @
102 @ CONTROL WORD *****
103 C---- OBJECT
104 @
105 @ ANAMO *****
106 K---- P1-HL Hot leg
107 @
108 @ ITYP0 FPAR0 ICMPO
109 @ 20 1.0 0
110 @
111 ----- NETWORK
112 @ S01 NI01
113 @ 0.0 1
114 @ 7.240
115 @
116 @----- JUNTYPES
117 @ S0 JTYP0 JTYPI0
118 @
119 ----- GEOMETRY
120 @ S0 Z0 D0 A0 V0 DEPO
121 @ 0.0 0.0 0.75 0.0 0.0 0.0
122 @ 7.240 0.0 0.75 0.0 0.0 0.0
123 @
124 ----- FRICTION
125 @ ITPMO ALAMO ROUO
126 @ %ITPMO-P% 0.02 1.D-5
127 @ SFO SDFJO ZFFJO ZFBJO
128 @ 0.0 0.0 2.59 2.59
129 @ 7.240 0.0 5.00 1.67
130 @
131 ----- DRIFT

```

Abb. 4.29 TFO Daten in ATHLET ASCII Datei

Die Verbindungsdaten, d. h. die Topologie der Objekte, werden einer ATHLET-Ausgabedatei (*.gr Datei) entnommen. In Abb. 4.29 ist in rot gekennzeichnet, dass das TFO 2 auf TFO 3 folgt und TFO 1 als Eingang besitzt.

```

C:\Simulators\testdata_athlet_31a\pid.rid.gr
0 1,0 2,0 3,0 4,0 5,0 6,0 7,0 8,0 9,0 10,0
190 @
191 @ K ANAMO IBO IEO SBPO SEPO IDFO IBI IBIL IEI IEI1
192 @ 1 PV-UP 0 0 0.00000E+00 0.00000E+00 0 0 0 0 0
193 @ 2 P1-HL 1 3 0.00000E+00 3.75000E-01 1 1 3 3 6
194 @ 3 P1-SG-UT 2 4 7.24000E+00 0.00000E+00 1 3 6 10 13
195 @ 4 P1-CL 3 5 2.41750E+01 3.75000E-01 1 10 13 16 19
196 @ 5 PV-DC 4 6 2.12700E+01 0.00000E+00 1 16 19 20 22
197 @ 6 PV-LP 0 0 0.00000E+00 0.00000E+00 0 0 0 0 0
198 @ 7 PV-COR1 6 1 0.00000E+00 0.00000E+00 1 22 24 25 1
199 @ 8 PO-SURGE 2 9 3.00000E+00 2.00000E-01 1 3 28 28 31
200 @ 9 PO-PRESS 8 0 1.50000E+01 0.00000E+00 1 28 31 31 0
201 @ 10 PO-LEAK 4 11 1.60000E+01 0.00000E+00 1 16 35 16 35
202 @ 11 PO-CONT 0 0 0.00000E+00 0.00000E+00 0 0 0 0 0
203 @ 12 S1-SG-DC-B 15 13 0.00000E+00 0.00000E+00 1 49 37 38 41
204 @ 13 S1-SG-RIS 12 14 1.08000E+01 0.00000E+00 1 38 41 44 47
205 @ 14 S1-SG-SEP 13 15 1.08000E+01 0.00000E+00 1 44 47 47 49
206 @ 15 S1-SG-DC-T 0 0 0.00000E+00 0.00000E+00 0 0 0 0 0
207 @ 16 S1-SG-DOM 15 17 0.00000E+00 0.00000E+00 1 49 51 51 54
208 @ 17 S1-AX-MSL 16 18 4.05000E+00 0.00000E+00 1 51 54 54 56
209 @ 18 S1-AX-CND 0 0 0.00000E+00 0.00000E+00 0 0 0 0 0
210 @ 19 S1-AX-FW 18 12 0.00000E+00 1.20000E+00 1 56 58 58 37

```

Abb. 4.30 ATHLET Objekttopologie in GR-Ausgabedatei

Mit diesen Daten werden die Verbindungen (Abb. 4.30) der TFO generiert. Die „Auto-trace“ Funktion in ATM erzeugt bei nicht zu komplexen Topologien meist ein kreuzungsfreies Layout. Im gegenwärtigen Zustand werden beim Import noch keine Geometrieobjekte für die TFO generiert. Die zugehörigen Daten sind in Tabellen gespeichert und können dort auch verändert werden.

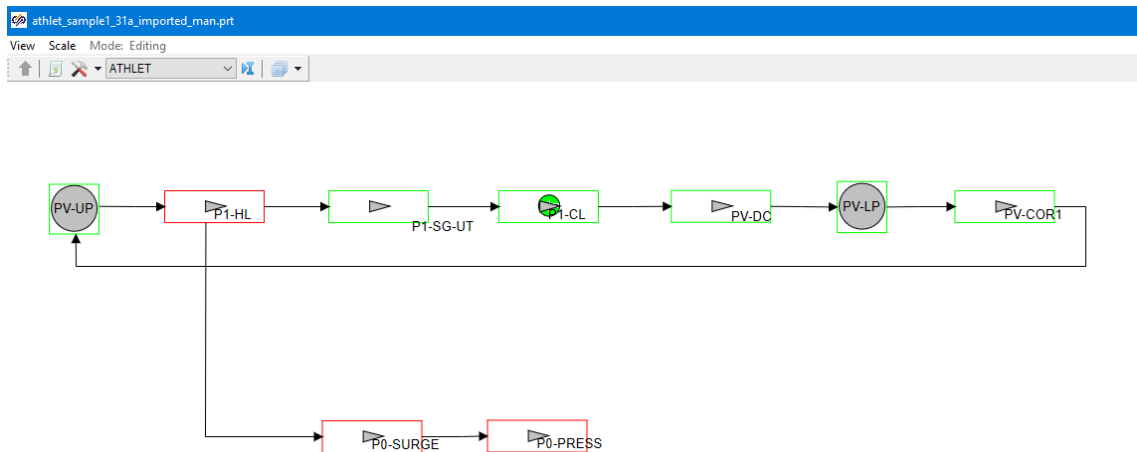


Abb. 4.31 Automatisch importierte TFO in ATM

4.1.3.2 Datenimport für ein TFO

Die zweite implementierte Funktion importiert ein TFO (Pipes, Branch, Cross-Connection-Object). Die ASCII-Objektdaten müssen dazu in den TFO-Editor von ATM für ein existierendes Objekt kopiert werden. Mit dieser Funktion können gezielt einzelne Objekte in ATM erzeugt oder geändert werden. Dies ist beispielsweise dann nützlich, wenn manuelle Änderungen aus dem mit ATM generierten Datensatz zurückgespielt werden sollen.

Die Daten des TFO müssen in der Ansicht „Source“ des TFO (Abb. 4.32) eingefügt werden (Copy/Paste). Im Anschluss werden die Daten in die Property-Tabellen des TFO übernommen und können gespeichert werden.

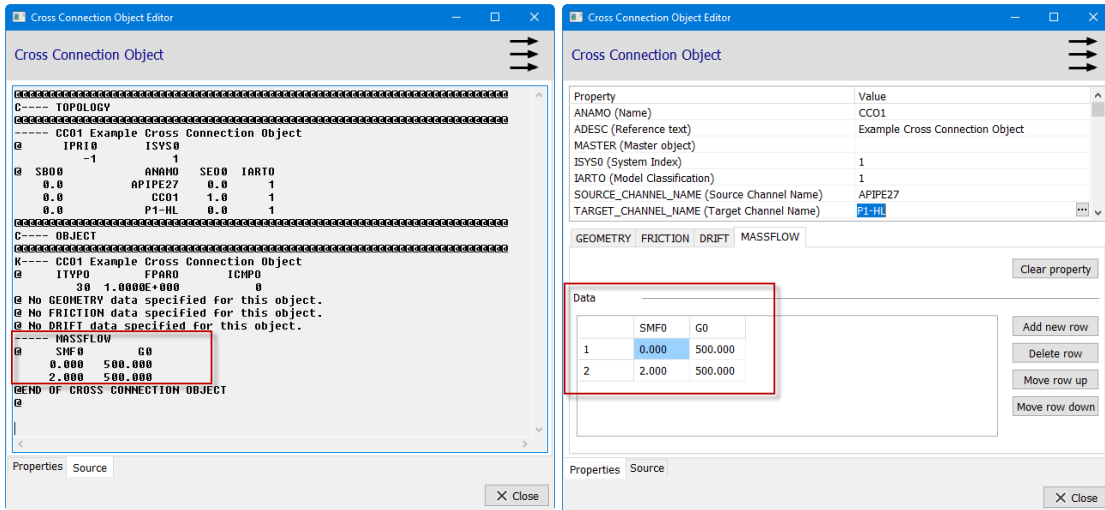



Abb. 4.32 Beispiel des Datenimports für ein CCO

4.1.3.3 Datenimport für weitere Daten

Mit dieser Funktion können Daten eines Eingabedatensatzes in ATM erfasst werden, die weder automatisch importiert oder interaktiv erstellt wurden. Damit ist dann möglich, den vollständigen Datensatz in ATM zu speichern und eine komplette Eingabe für ATHLET bereitzustellen. In den zentralen „Simulationseigenschaften“ eines Projekts in ATM wurden eine Reihe von Eingabefeldern (Abb. 4.33) eingeführt. Die Anzeige der Eigenschaften ist durch einen Klick auf das Symbol  möglich.

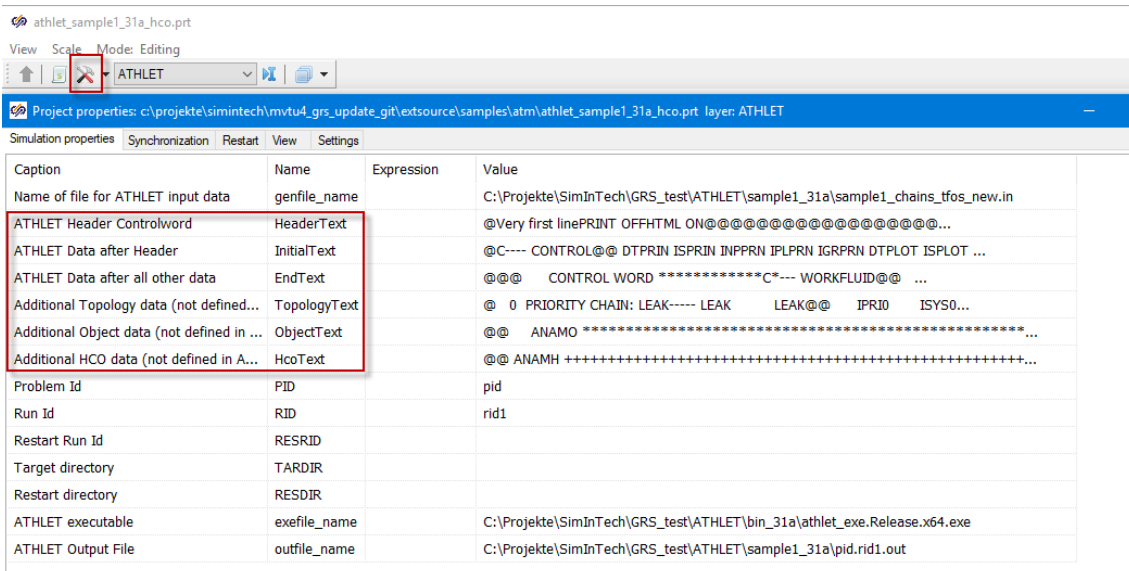
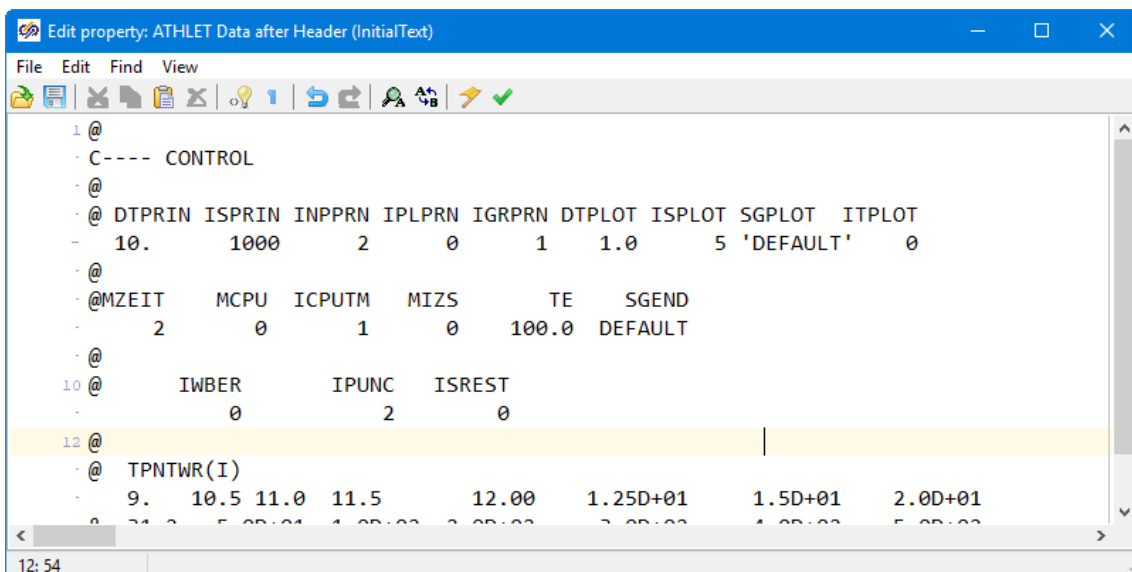


Abb. 4.33 Eingabefelder für ASCII-Datenimport (Simulation properties)

Die Daten dieser Felder, z. B für InitialText (siehe Abb. 4.34), werden bei der Erzeugung der Eingabedaten berücksichtigt und mit den interaktiv definierten Objekten in der folgenden Reihenfolge verwendet:

1. HeaderText
2. InitialText
3. Interaktiv definierte Prioritätsketten
4. TopologyText
5. Interaktiv definierte TFOs
6. ObjectText
7. Interaktiv definierte HCOs
8. HcoText
9. EndText

Dieses Konzept ermöglicht es, einen vorhandenen ASCII-Eingabedatensatz komplett zu importieren und die interaktive Modellierung mit ATM-Objekten schrittweise auszubauen. Die interaktiv modellierten Objekte müssen in den Eingabefeldern gelöscht werden, um eine Doppeldefinition zu vermeiden.



```
1 @
- C---- CONTROL
- @
- @ DTTPRN ISPRIN INPPRN IPLPRN IGRPRN DTPLOT ISPLOT SGPLOT ITPLOT
- 10. 1000 2 0 1 1.0 5 'DEFAULT' 0
- @
- @MZEIT MCPU ICPUTM MIZS TE SGEND
- 2 0 1 0 100.0 DEFAULT
- @
10 @ IWBER IPUNC ISREST
- 0 2 0
12 @
- @ TPNTWR(I)
- 9. 10.5 11.0 11.5 12.00 1.25D+01 1.50+01 2.00+01
```

Abb. 4.34 Beispiel für Eingabedaten im ATM zur ATHLET Steuerung

4.1.4 Erzeugung und Test der Eingabedaten

Nach der Erstellung aller notwendigen Eingabedaten in ATM können diese im von ATHLET benötigten Format als ASCII-Datei ausgegeben werden. Ein Test der Daten

mittels Durchführung einer ATHLET Simulation kann im Anschluss durchgeführt werden. Diese Funktionen können über das ATHLET-Menü (Abb. 4.13) verwendet werden. Die dafür notwendigen Daten werden im Dialog für die „Simulation properties“ definiert (Abb. 4.35).

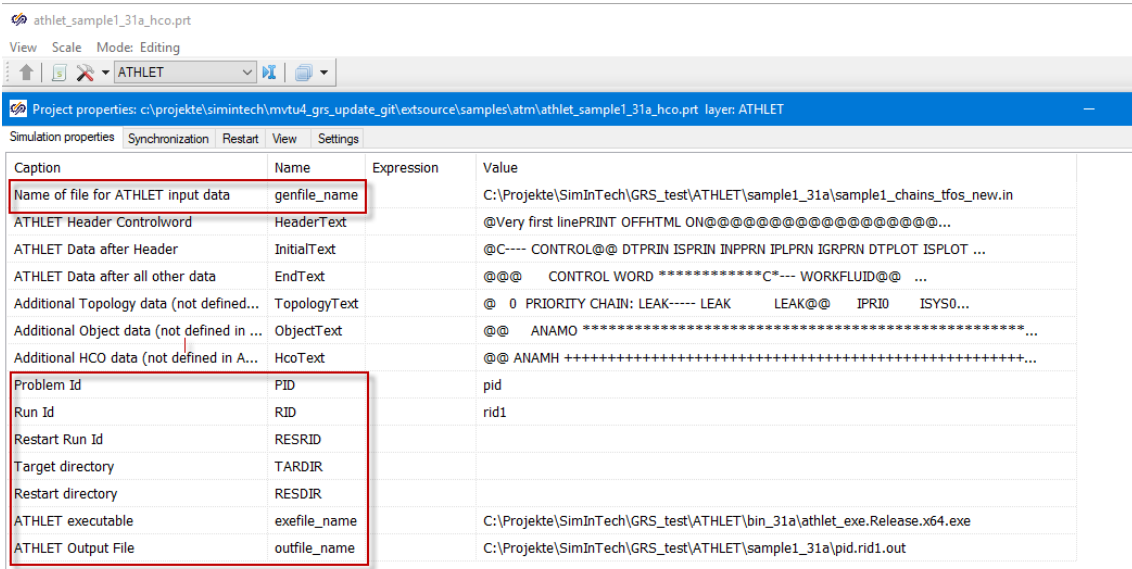


Abb. 4.35 Daten für Simulationsausführung (Simulation properties)

Die Bedeutung der Daten ist nachfolgend erläutert:

Genfile_name	Name der ATHLET Eingabedatei
PID, RID, RESRID, TARDIR, RESDIR	ATHLET Startparameter
Exefile_name	Vollständiger Pfad für ATHLET-Executable
Outfile_name	ATHLET Ausgabedatei

Beim Erzeugen der Eingabedaten werden alle interaktiv definierten Objekte berücksichtigt. Die Daten werden, soweit möglich, geprüft und es werden entsprechende Hinweise ausgegeben (Abb. 4.36).

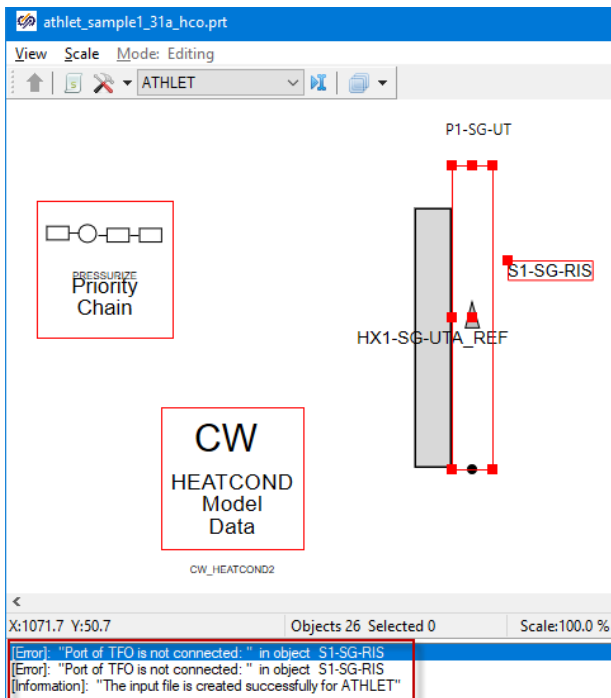


Abb. 4.36 Hinweise im „Message Window“ beim Erzeugen der Eingabedaten

Ein detaillierter Test ist aber nur mit einem ATHLET Simulationslauf möglich. Dieser wird in einem CMD-Fenster durchgeführt und die Ergebnisse, d. h. die Ausgabedatei der Simulation, können in ATM direkt angezeigt werden. Auf diese ist eine schrittweise Verbesserung der Daten einfach durchführbar.


4.1.5 Modellierung parametrischer Objekte

Mit der Bereitstellung parametrischer Objekte, deren Eigenschaften, wie Geometriedaten oder Diskretisierung, durch eine zentrale, einfache Datenvorgabe geändert werden können, kann der Benutzer Modifikationen im Datensatz leichter, schneller und weniger fehleranfällig durchführen.

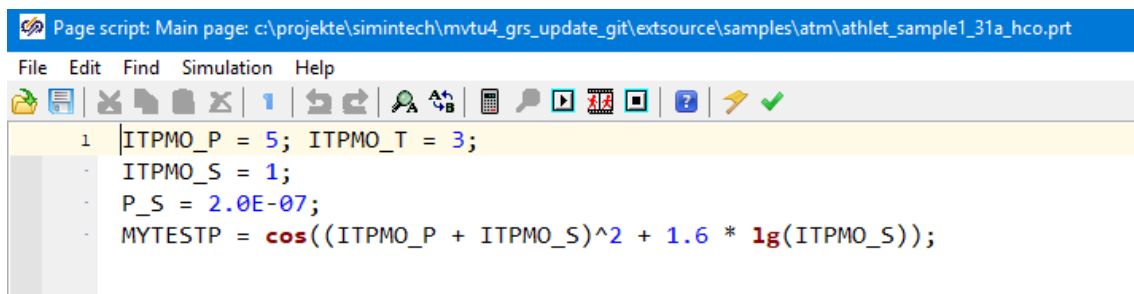
4.1.5.1 Parameter

Mit der Bereitstellung von Parametern, also Variablen, für die Werte der Eigenschaften von Objekten, wie beispielsweise Modeloptionen oder Anfangszustände, kann der Benutzer Modifikationen in den Daten an zentraler Stelle effizient durchführen. Insbesondere bei der Verwendung eines Parameters in mehreren Objekten oder zur Einstellung einer Simulationsvariante ist dieses Vorgehen sehr einfach und intuitiv. Bereits die

Standard ATHLET Eingabe stellt eine entsprechende Option mit dem „PARAMETERS“-Eingabeblock /LER 16/ bereit.

In ATM ist zur Verwendung von Parametern in Modellierung ebenfalls eine entsprechende Option implementiert. In jedem Arbeitsblatt gibt es neben der Diagrammdarstellung (Objektworkspace) auch noch einen zugeordneten Parameterworkspace (Abb. 4.37). Dieser wird durch einen Mausklick auf das Symbol  am Kopf des Arbeitsblatts angezeigt. Dort können Parameter mit Namen und Wert definiert werden.

Neben einer einfachen Wertzuordnung sind zusätzlich auch komplexere Parameter-Operationen möglich, wie z. B. Kombination mehrerer Parameter in arithmetischen Ausdrücken.

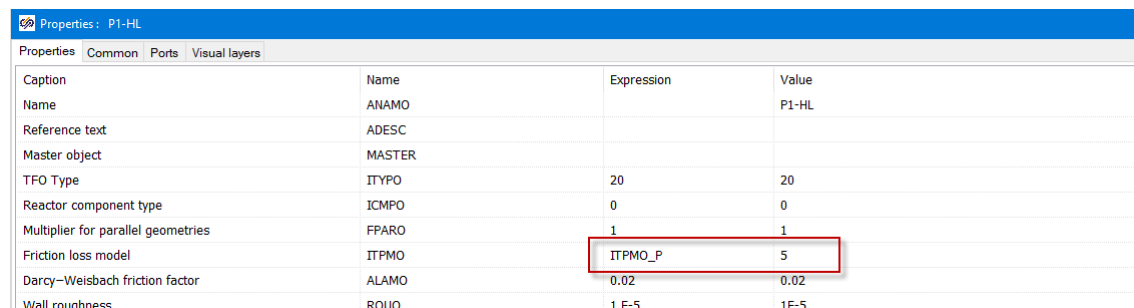


```

1 | ITPMO_P = 5; ITPMO_T = 3;
  | ITPMO_S = 1;
  | P_S = 2.0E-07;
  | MYTESTP = cos((ITPMO_P + ITPMO_S)^2 + 1.6 * lg(ITPMO_S));
  
```

Abb. 4.37 Workspace für die Definition von Parametern

Die dort definierten Parameter können über ihren Namen in den Property Dialogen der Objekte referenziert werden, indem sie in der Spalte „Expression“ eingetragen werden. Das in Abb. 4.38 gezeigte Beispiel zeigt die Verwendung des Parameters „ITPMO_P“ zur Einstellung des Modells zur Berechnung des Reibungsdruckverlusts in einem TFO.



Properties	Common	Ports	Visual layers
Caption	Name	Expression	Value
Name	ANAMO		P1-HL
Reference text	ADESC		
Master object	MASTER		
TFO Type	ITYPO	20	20
Reactor component type	ICMPO	0	0
Multiplier for parallel geometries	FPARO	1	1
Friction loss model	ITPMO	ITPMO_P	5
Darcy-Weisbach friction factor	ALAMO	0.02	0.02
Wall roughness	ROUO	1.E-5	1E-5

Abb. 4.38 Verwendung von Parametern im Property Dialog

Die aktuellen Werte der Parameter können in der Spalte „Value“ kontrolliert werden.

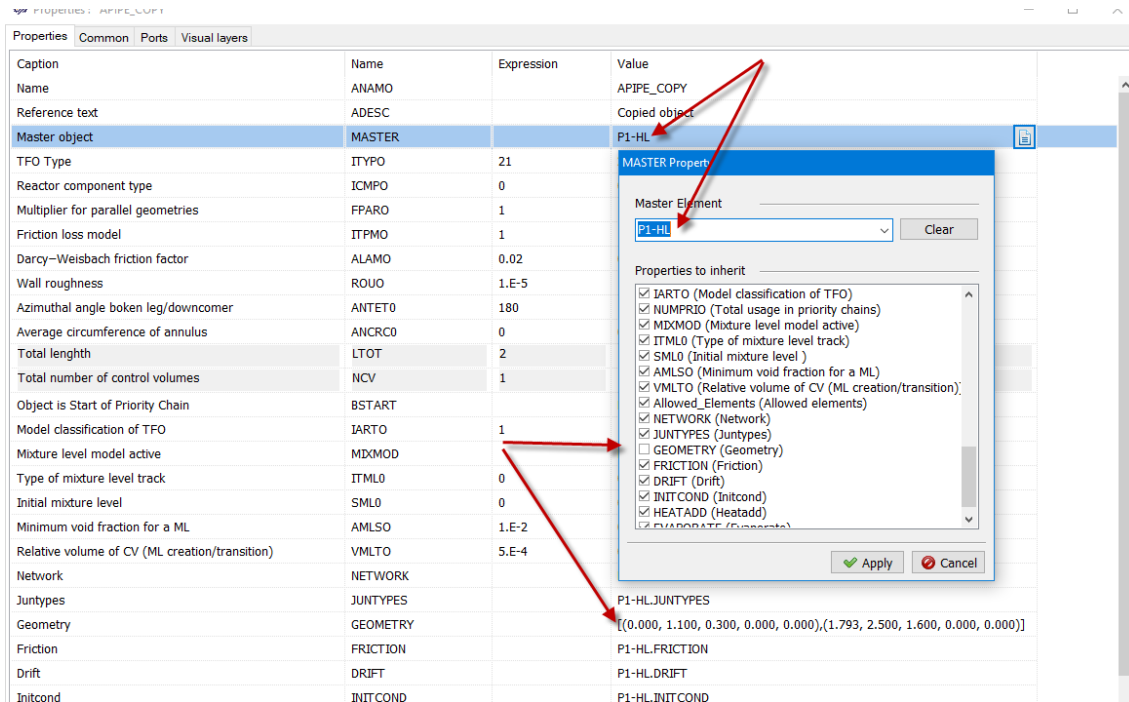


Abb. 4.40 TFO als Kopie eines Masterobjekts

In der Property Master kann der Name eines „Master“-TFO vorgegeben werden. Der Name des Masterobjekts („Master Element“) kann im „MASTER Property“ Dialog in einer aus eine Liste von TFOs ausgewählt werden. Ist ein Masterobjekt gewählt, werden alle Daten der Objektkopie vom Masterobjekt vererbt, die im „MASTER Property“ Dialog mit in den Check Boxes markiert sind. Nicht vererbte Daten können im Property-Editor der Objekte wie üblich spezifiziert werden und weisen damit der Objektkopie eigene Daten zu, z. B. für die Geometrie. In Abb. 4.41 ist der entsprechende ASCII-Input für ATHLET einer Objektkopie für ein TFO zu sehen.

```

- @ ANAMO *****
- K---- APIPE_COPY Copied object
130 @ ITYPO FPARO ICMPO
- 21 1.0000E+000 0
- COPY P1-HL
- ---- GEOMETRY
- @ SGO Z0 A0 V0 DEP0
- 0.000 1.100 0.300 0.000 0.000
- 1.793 2.500 1.600 0.000 0.000

```

Abb. 4.41 Kopie eines TFO in ATHLET

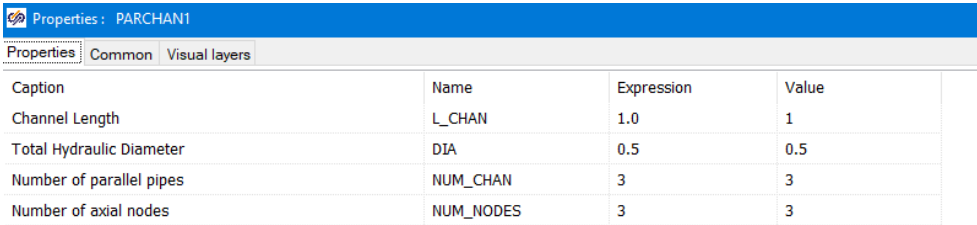
4.1.5.3 Makro Objekte

Makro-Objekte sollen die Modellierung von komplexen und öfter benötigten thermohydraulischen Komponenten wie Dampferzeuger oder RDB vereinfachen. Die entwickelte Methode nutzt ATHLET TFOs, um diese in einem geeigneten Netzwerk zu verbinden und kann typische Daten, wie Geometrie und Diskretisierung, entsprechend vorgegebener Werte, zu variieren. Zusätzlich zu den zuvor beschriebenen Parametern und Objektkopien (Master Objekte), werden mit Hilfe von „Scripting“ dynamisch Fluid- und Topologieobjekte (TFOs, PCs) erzeugt. Dadurch können die Makro-Objekte nicht nur hinsichtlich ähnlicher Daten (z. B. Länge) angepasst werden, sondern auch die Detaillierung bei der Diskretisierung variiert werden. Die Skriptsprache von SimIntech enthält sowohl übliche Befehle zur Programmsteuerung (Schleifen, Bedingungen, etc.) als auch vordefinierte Funktionen zum Zugriff und zur Modifikation von Objekten.

Die Methode wird im Folgenden an einem Beispiel erläutert. In vielen Anwendungen, z. B. für Behälter in Versuchsanlagen oder im RDB, ist ein Zylinderobjekt nützlich, mit den folgenden einstellbaren Parametern:

- Durchmesser
- Länge
- Anzahl der Ringe in radialer Richtung
- Anzahl der Zellen in axialer Richtung

Die Ringe im Zylinder lassen sich in ATHLET mit parallelen Pipes modellieren, die untereinander mit „Cross Connection Objects“ verbunden sind. Daher wurde ein entsprechendes Makro-Objekt (Abb. 4.42) in ATM erstellt, das diese Parameter als modifizierbare Properties enthält.



Caption	Name	Expression	Value
Channel Length	L_CHAN	1.0	1
Total Hydraulic Diameter	DIA	0.5	0.5
Number of parallel pipes	NUM_CHAN	3	3
Number of axial nodes	NUM_NODES	3	3

Abb. 4.42 Makro-Objekt für Parallelkanal-Modellierung eines Zylinders

Wenn man die Anzahl der parallelen Pipes (NUM_CHAN) für die Simulation anpasst, werden automatisch die dafür notwendigen Objekte generiert. In Abb. 4.43 sind als

Beispiel eine Simulation eines Zylinders mit drei parallelen TFO gezeigt. Dabei sind MP2 und MP3 Kopien des TFO MP1. Diese bilden einen Zentralkanal und zwei konzentrische Ringe ab, die mit CCO verbunden sind. Will man die Anzahl der Kanäle nachträglich ändern, ist es zurzeit nötig, die Kopien des Master-Pipes manuell zu löschen.

Beim Ändern der geometrischen Parameter werden die Werte in den parallelen TFO (MP1, MP2, MP3) automatisch angepasst.

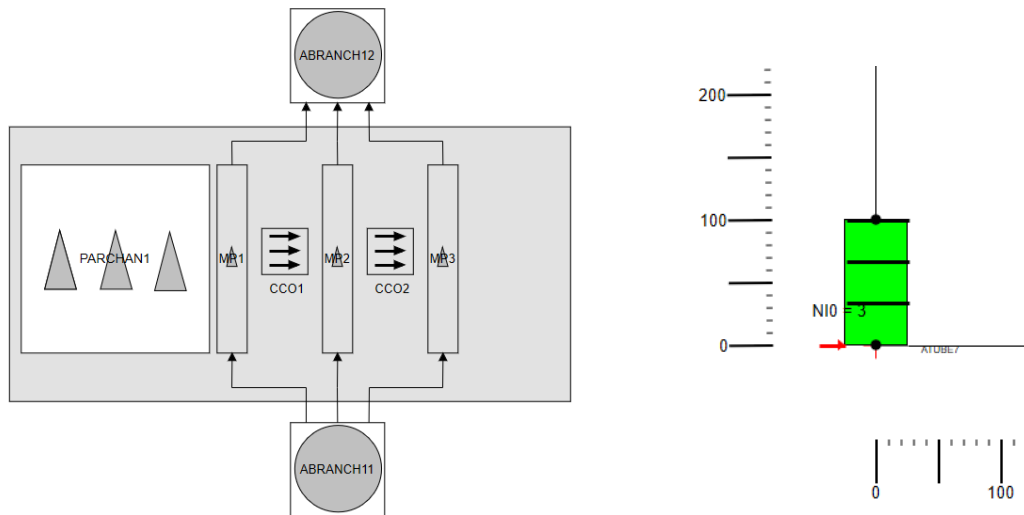


Abb. 4.43 Simulation eines vertikalen Zylinders mit parallelen TFO

Prinzipiell kann dieses Vorgehen zur Simulation beliebiger Komponenten durch den Benutzer erweitert und angepasst werden. Somit lässt sich die Bibliothek von Makro-Objekten schrittweise ausbauen. Allerdings hat sich in der Testphase herausgestellt, dass der gegenwärtige Funktionsumfang der Skriptsprache (z. B. der fehlende Zugriff auf Subworkspaces, Löschen von Objekten) und auch Probleme in der Implementierung, die Erstellung von komplexen Makro-Objekten einschränken oder unmöglich machen. In den künftigen Entwicklungsschritten sind daher einige grundsätzliche Verbesserungen im Scripting notwendig, um Objekte mit vielen Parametern und variabler Diskretisierung benutzerfreundlich realisieren zu können.

4.1.6 Datenexport für Visualisierung

Die in ATM erstellten Workspaces zur Simulation von Fluid-Systemen, wie die Darstellung der Topologie und der TFO_Geometrien sollen auch in ATLASneo verwendbar sein. Dies ist nützlich, um in der Simulation grafische Information zu verwenden

Eingabedaten bereitzustellen oder die Darstellungen bei Bedarf mit dynamischen Informationen aus der Simulation zu ergänzen.

Das für ATLASneo verwendete Bildformat ist das SVG-Format. Daher war es nötig, in ATM einen Diagrammexport in diesem Bildformat zu implementieren. Dies wurde durch eine geeignete Transformation der grafischen Primitive, wie Linie, Rechteck, Kreis usw., und der zugehörigen Daten, in die entsprechende SVG-Syntax realisiert. Prinzipiell wurden dafür geeignete Transferfunktionen erstellt, wie z. B.:

```
procedure      TLine.SaveToSVG (fVisibleFlag: boolean;const aStrings:
TStringList;const aIdPrefix: string);
```

Mit diesen Prozeduren können alle grafischen Objekte auf einem Workspace in ATM zu einer entsprechenden „Group“ in SVG exportiert werden. Der Export ist über das Menü „Service“ (Abb. 4.44) aus der Hauptbedienleiste von ATM ansteuerbar.

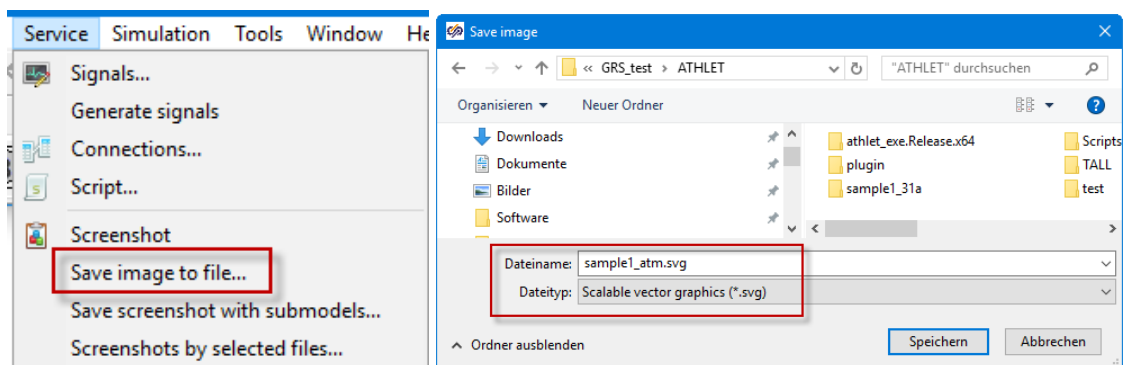


Abb. 4.44 Export eines Workspaces aus ATM ins SVG-Format

Das exportierte Bild kann dann in einem geeigneten Bildeditor, wie z. B. Inkscape /INK 17/, überprüft (Abb. 4.45) und bei Bedarf geändert werden.

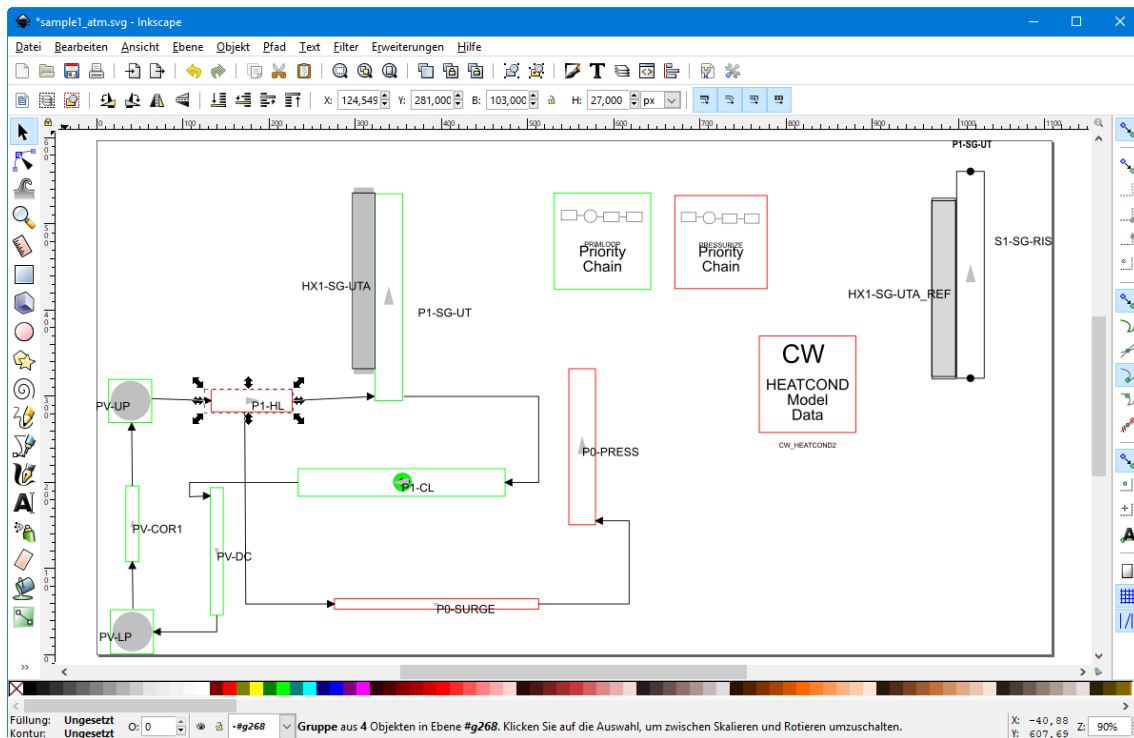


Abb. 4.45 Primärkreislaufdarstellung als SVG-Export aus ATM

Im letzten Schritt kann dann die erstellte SVG-Grafik auch in ATLASneo im „EIDOS“-Gadget angezeigt werden. Das Bild kann dort auch mit den dynamischen Simulationsdaten animiert werden. Die Verknüpfung der Bildelemente und deren Eigenschaften mit den dynamischen Daten ist noch nicht vollständig entwickelt. In Abb. 4.46 ist die Darstellung des TFO „P1-HL“ mit der Fluidtemperatur aus der ATHLET-Simulation verknüpft. Die Farbe des Elements ändert sich entsprechend einer Farbskala und dem aktuellen dynamischen Wert. Dieser kann numerisch in einem Tooltip eingeblendet werden. Dies ist ein Beispiel für eine mögliche Dynamisierung, andere wie die Anzeige des numerischen Werts oder Farbumschläge bei Überschreiten eines Limits, sind aber ebenfalls denkbar. Eine Automatisierung der Dynamikdefinition im SVG-Export von ATM kann einfach ergänzt werden, sobald die Art der benötigten Bilddynamikinformation mit den Anwendern abgestimmt ist.

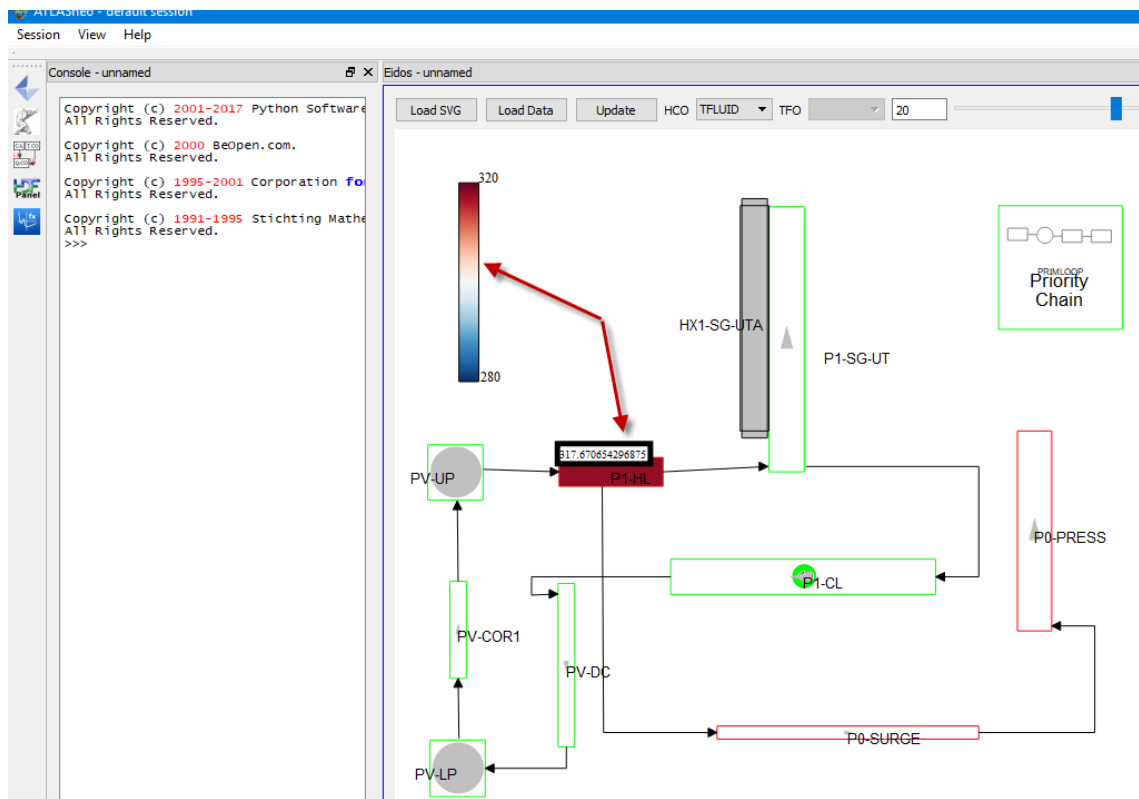


Abb. 4.46 ATM Workspace mit dynamischen Daten in ATLASneo

Prinzipiell ist es für einige Anwendungen, wie die Simulation eines Behälters mit vielen parallelen Kanälen, auch wünschenswert, eine entsprechende 3D-Visualisierung zu erzeugen, da Schnitte stets nur Teile der benötigten Daten zeigen können. Im Vorhaben RS1199 /VOG 15/ wurden geeignete Methoden auf Basis von FreeCAD und ParaView entwickelt. Die Erzeugung der 3D-Geometrie für die TFO wurde manuell mit FreeCAD durchgeführt und kann dann in einem für ParaView lesbaren Format zur Verfügung gestellt werden. Da FreeCAD mit Python-Skripten erweitert werden kann, kann diese Arbeit durch den Export von geometrischen Daten und der Objekttopologie, die in ATM bereits vorliegen, unterstützt werden. Eine generelle, vollständige Automatisierung ist allerdings nicht möglich, da eine Vielzahl notwendiger Daten, wie die genaue geometrische 3D-Form einzelner Zellen, nicht in ATM vorliegt. Bei der Implementierung dieser Anwendung wurde ein Datenexport auf XML-Basis genutzt, der alle Daten enthält, die in ATM für die Beschreibung eines Systems vorhanden sind. Die in Abschnitt 4.1.8.2 entwickelten Funktionen wurden so erweitert, dass sie auch die Daten aus ATM-Objekten verarbeiten können. Da die Funktionen in Python implementiert sind, können diese in FreeCAD direkt aus der Konsole verwendet werden, um die ATM-Daten für die 3D-Modellierung verfügbar zu machen.

Gegenwärtig ist aber wegen erheblicher Änderungen in der aktuellen FreeCAD-Version eine Überarbeitung der aus RS1199 vorhandenen Werkzeuge für die 3D-Modellierung für ATHLET nötig. Im Anschluss daran können die Zugriffsfunktionen auf ATM-Daten ausführlicher eingesetzt und getestet werden.

4.1.7 Qualitätssicherung und Dokumentation

Begleitend zur Entwicklung von ATM wurden die Entwicklungsschritte Fachleuten aus der ATHLET-Anwendung regelmäßig vorgestellt. In der Diskussion ergaben sich regelmäßig Hinweise für Verbesserungspotential oder Prioritäten in der Realisierung von Funktionen. Mit diesem Vorgehen konnte nicht bedarfsgerechte oder fehlerhafte Entwicklung frühzeitig erkannt und unnötiger Aufwand vermieden werden.

Aus unterschiedlichen Gründen, besonders wegen unvollständiger Implementierung der ATHLET Objekte, wurde ATM noch nicht für eine Anwendung in der Datensatzerstellung für eine reale Anlage eingesetzt. Als Alternative wurde die interaktive Modellierung eines Beispieldatensatzes für ATHLET für einen DWR als Test für die Anwendbarkeit genutzt. Dabei wurde das in Abschnitt 4.1.4 erläuterte Verfahren genutzt, Teile des ASCII-Datensatzes mit ATM-Objekten zu ersetzen. Es konnte gezeigt werden, dass das in ATM erstellte Modell identische Rechenergebnisse wie bei Verwendung der ursprünglichen Daten erzeugt. Damit kann für diesen Fall belegt werden, dass die in ATM implementierte Methode korrekte Eingabedaten für eine ATHLET Simulation erzeugen kann.

Die durchgeführten Entwicklungsschritte wurden mit Hilfe des Projektmanagementsystems Trac /TRA 18/ geplant und dokumentiert. Dieses lizenzfrei einsetzbare System vereint verschiedenste Werkzeuge zur Verwaltung von Projekten aller Art und stellt diese den Entwicklern als webbasierte Anwendung zur Verfügung. Den zentralen Bestandteil bildet hierbei das Wiki-System, welches sich durch seine einfache Syntax hervorragend zur schnellen Dokumentation von Entwicklungsschritten oder Zwischenergebnissen eignet. Wie für Wiki-Systeme typisch, können Beiträge und Änderungen direkt im Browser als schlicht formatierte Texteingaben eingegeben werden. Beim Absenden werden diese in fertige HTML-Seiten umgewandelt und stehen anderen sofort zur Verfügung.

Sowohl erkannte Programmfehler als auch die Liste geplanter Erweiterungen werden im integrierten Bugtrackingsystem als Tickets erfasst und verwaltet. Die Bearbeitung eines Problems und die zu dessen Lösung durchgeführten Maßnahmen können direkt im jeweiligen Ticket dokumentiert werden und sorgen durch eine Verlinkung zur

entsprechenden Revision in der Quellcode-Versionsverwaltung für eine gute Nachvollziehbarkeit von Quellcodeänderungen.

Durch das in Trac vorgesehene Zusammenfassen von Tickets zu Meilensteinen wird der zeitliche Verlauf des Projekts in Form eines Projektplans organisiert. Damit können ungelöste Probleme im Auge behalten und die zum Erreichen eines Meilensteins notwendigen Arbeiten geplant werden.

Die entwickelten Programme werden in einem System zur Versionsverwaltung von Dateien verwaltet. Damit können die Änderungen anhand von Revisionen nachvollzogen werden und beliebige Versionen der Programme wiederhergestellt werden. Zu Beginn des Vorhabens wurde die zentrale Versionsverwaltung Subversion /SVN 18/ eingesetzt. Mittlerweile wird die dezentrale Verwaltung Git /GIT 18/ verwendet, die lokale Kopien eines Repositories unterstützt. Die Umstellung erfolgte, um die neuen Versionen der Basissoftware SimInTech, die ebenfalls mit Git verwaltet wird, einfacher übernehmen zu können.

Die interaktive Hilfe für ATM wurde auf Basis von HTML-Seiten erstellt, die über entsprechende Links („TopicID“) kontextsensitiv aus dem Programm angesteuert werden können. Die Startseite für die Hilfe wird in ATM spezifiziert (Abb. 4.47), die TopicID wird beim entsprechenden Objekt gesetzt.

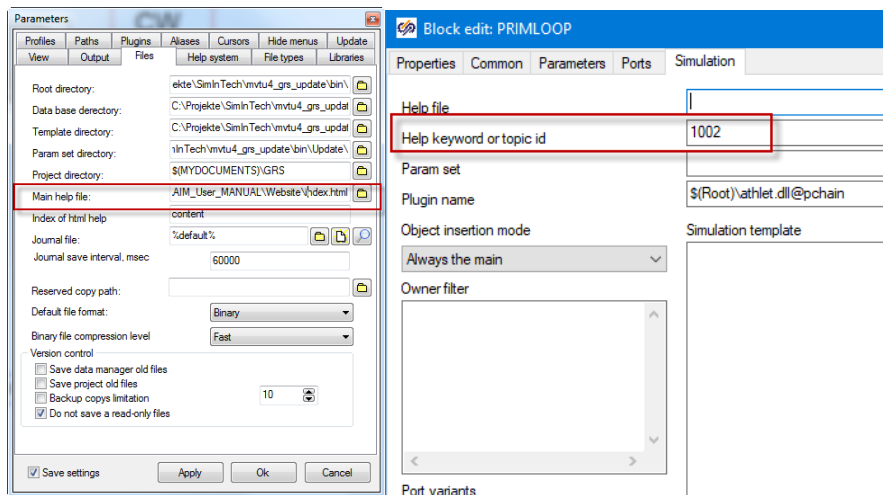


Abb. 4.47 Anbindung der interaktiven Hilfe in ATM

Beim jeweiligem Objekt wird dann die entsprechende HTML-Seite direkt im Browser angezeigt.

← → ↻ Datei | file:///C:/Projekte/SimInTech/mvtu4_grs_update_git/extsource/ATM_User_MANUAL/Website/index.html

Apps ★ Bookmarks 📄 Google Lesezeichen 📄 GRS Portal 📄 Google Übersetzer 🌱 Unlesbare kyrillische 🌐 ATLAS User's Area 🛠 TeamForge : ATLAS 🚀 SSDev 🚀 ATHLET 🚀 ATLAS_neo 📄

Inhalt Index Suchen

Inhaltsverzeichnis


- AIM User's Manual
 - Overview
 - Installation
 - Getting started
 - User Interface
 - AGM Usage
 - ATM Usage
 - Overview
 - Generation of a new thermal hydraulic model
 - How to start
 - ATHLET objects toolbars
 - General
 - General elements
 - ATHLET Elements toolbar
 - pipe
 - Branch
 - Single junction pipe
 - Fill (Single junction pipe)
 - Time dependent volume
 - Cross connection object
 - Priority chain (PC) object
 - Heat conduction object
 - Basic piping objects
 - Control word objects
 - ATHLET menu
 - Parametric Objects

Copyright 2009, Ihre Firma

Time dependent volume

The p-h boundary model (also referred as 'time dependent volume') is a CV-related model. It is used for the simulation of a pressure-enthalpy boundary at the edge or within the TFD system. The model can be applied for a branch or for a pipe with just one or several CVs. If the pipe contains several CVs, all CVs are defined by the same set of GCSM signals, i.e. they all have identical thermodynamic states.

The p-h boundary CV can be coupled with all types of junctions and CVs. Keep in mind that the momentum equation of the junctions is always solved between the centers of the connected CVs, also if the CV is a p-h boundary CV. Since the flows in the junctions depend - among others - on length-weighted average values of the CV properties, the geometry of the p-h boundary CVs is of importance. Thus, specify reasonable geometry data, comparable to the adjacent CVs. If a p-h boundary for a discharge is represented, enter a short length ($\approx 0.01\text{m}$) and a CSA close to that of the connected junction(s).



During the transient calculation, the state of the fluid in the CV is completely defined by the pressure and total enthalpy specified by GCSM signals. These signal can be specified at the end of the prop

GCSM signal fluid pressure	SGPRES		
GCSM signal fluid enthalpy	SGENTH		
No. of NC gases	NCNO	0	0
Name of NC gas	AGAS		
SXQM	SXQM		
SXPGAS	SXPGAS		
STEMP	STEMP		
SXMGAS	SXMGAS		

In a p-h boundary object, any gas or gas mixture can be specified provided the corresponding gases are simulated in the related TFD system. You can find further information about how to use it in [ATHLET input description](#)

Abb. 4.48 Beispielseite für interaktive Hilfe in ATM

Im Allgemeinen wird ein kurzer Überblick zum jeweiligen Objekt gegeben, teilweise mit Hinweisen zu Verwendung oder Bedienung. Für weitere Details, insbesondere zur Bedeutung der Eingabeparameter für das Objekt, ist die ATHLET Eingabebeschreibung verlinkt. Da diese gegenwärtig keine Sprungmarken (ID) enthält, muss mit der Suchfunktion des Browsers zur entsprechenden Stelle navigiert werden. Dieses Vorgehen vereinfacht jedoch die Aktualisierung der Hilfe in ATM, falls sich die Eingabebeschreibung von ATHLET ändert.

Die zuvor für AGM verwendete Hilfe auf Basis von „Microsoft Compiled HTML Help“ wurde ebenfalls auf HTML-Seiten umgestellt und überarbeitet. Dies hat den Vorteil, dass in der Linux-Version die interaktive Hilfe ohne Verwendung von zusätzlicher Software verfügbar ist.

Die Erstellung der Hilfeseiten erfolgt mit FastHelp /FAH 18/, das auch eine Erzeugung eines äquivalenten Benutzerhandbuchs (User Manual) im PDF-Format unterstützt

/VOG 18/. Damit ist die Konsistenz von interaktiver Hilfe und Benutzerhandbuch sehr einfach sichergestellt.

4.1.8 Erweiterung der Leittechnikmodellierung mit AGM

Der „ATHLET GCSM Modeller“ (AGM) wurde im Vorläufervorhaben RS1199 /VOG 15/ entwickelt und wird bereits zur Modellierung von Reaktorleittechnik in Anwendungsprojekten verwendet. Einschränkungen bei der Anwendung ergeben sich bisher durch die fehlende Lauffähigkeit unter Linux-basierten Systemen.

Dies ist durch die verwendete Entwicklungsumgebung Delphi /EMB 15/ begründet, die nur Microsoft Windows unterstützt.

Außerdem liegt eine große Zahl umfangreicher Leittechnikmodelle vor, die im bisher von ATHLET verwendeten G2-Modellierer erstellt wurden und in dessen proprietären Binärformat gespeichert sind. Langfristig ist es sinnvoll nur AGM zur Modellierung einzusetzen. Zum Transfer der vorliegenden G2-basierten Modelle fehlt daher ein Verfahren, diese Modelle automatisiert in AGM übernehmen zu können.

4.1.8.1 Linux-Portierung von AGM

AGM ist als ein Erweiterungsbaustein (Plugin) der Basissoftware „SimInTech“ in Form von dynamischen Laufzeitbibliotheken (DLLs) unter Windows mit Delphi und dem Intel Fortran Compiler entwickelt worden.

Im Rahmen des Vorhabens wurde eine neue Version der Basissoftware eingesetzt, die sowohl Windows als auch Linux unterstützt. Diese Version ist eine Portierung der Delphi Version auf den Free Pascal Compiler /FPC 15/, der als OpenSource frei verfügbar ist. Zur Erzeugung der Bedienoberflächen nutzt der Compiler die plattformübergreifenden GUI-Bibliotheken Qt /QTC 17/ oder GTK. Diese unterscheiden sich von den in Delphi verwendeten und erheblicher Aufwand entsteht daher bei der Portierung der Bedienoberflächen von AGM.

Zur Programmentwicklung sind verschiedene Entwicklungsumgebungen nötig, unter Windows Delphi Tokyo (Embarcadero) und unter Linux CodeTyphon/FreePascal (Pilot-Logic). Letztere ist auch unter Windows verfügbar und wurde dort auch getestet. Leider lassen Funktionsumfang und Einschränkungen, besonders beim Debuggen der

Programme, die alleinige Verwendung dieser Umgebung nicht zu und erfordern die weitere Verwendung von Delphi als primäres Entwicklungswerkzeug. Neue Funktionen in der Basissoftware sind ebenfalls zunächst nur in der Windows Version verfügbar. Im Rahmen eines Supportvertrags werden diese dann in größeren Zeitabständen auch in die Linux-Version übernommen.

Die neue Version von SimInTech wurde intensiv auf ihre Funktionalität im Vergleich zur bisher verwendeten Windows-Variante getestet. Dazu wurden Software und benötigte Komponenten, insbesondere auch die notwendige Entwicklungsumgebung „CodeTyphon“, auf einem Linux-System installiert. Mit dieser Umgebung konnte SimInTech lauffähig kompiliert werden. Dies war deshalb aufwendig, da SimInTech neben den Standardbibliotheken des Pascal Compilers noch weitere Erweiterungskomponenten, z. B. zur Erzeugung der X/Y-Diagramme, verwendet. Diese müssen, in der passenden Version, ebenfalls unter CodeTyphon erzeugt und eingebunden werden.

In einem weiteren Schritt wurden unter Linux auch AGM und ATM als Laufzeitbibliotheken (shared libraries) kompiliert. Dazu war eine Reihe von Änderungen im Quellcode der Windowsversion, insbesondere bei den Schnittstellen zur Basissoftware und in den Bediendialogen notwendig. Die unvermeidbaren Erweiterungen an der Basissoftware wurden so gering wie möglich gehalten und durch Präprozessoranweisungen (ifdef) gekennzeichnet, damit künftige Updates des Herstellers auf einfache Weise verwendet werden können. Gleichzeitig wurden die Schnittstellen in AGM und ATM vereinheitlicht, so dass die gleichzeitige Verwendung beider Module unter einer gemeinsamen Bedienoberfläche, mit der neuen Bezeichnung ATHLET Input Modeller (siehe Abb. 4.49), möglich ist.

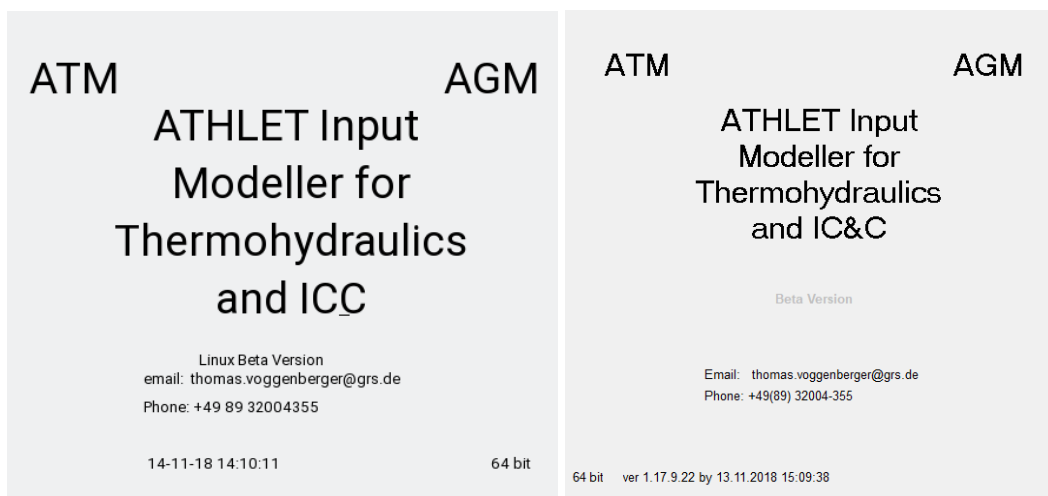


Abb. 4.49 Linux und Windows Version des „About“-Fensters

Da die Benutzerdialogmasken zwischen Delphi und FreePascal nicht kompatibel sind, wurden diese für Linux neu erstellt und getestet. Aus diesem Grund ist es sinnvoll, bei künftigen Erweiterungen, die Verwendung neuer Dialogmasken auf ein Minimum zu beschränken, da diese in beiden Versionen unterschiedlich sind und gepflegt werden müssen.

Nach der Implementierung von AGM unter Linux wurde die Funktionalität auf dieser Plattform intensiv getestet, besonders die Systemerstellung, die Datenausgabe für ATHLET und die Systemsimulation. Alle Fehler und Abweichungen wurden untersucht und weitgehend beseitigt oder vereinheitlicht. Für die Systemsimulation wird neben internen Funktionen in AGM auch eine FORTRAN-Bibliothek verwendet, die die gleichen Funktionen wie in ATHLET enthält. Diese wurde daher unter Linux mit dem GNU-Compiler übersetzt und eingebunden. Getestet wurde die Simulation u. a. mit einem System mit 2-Punkt-Regelung (Abb. 4.50).

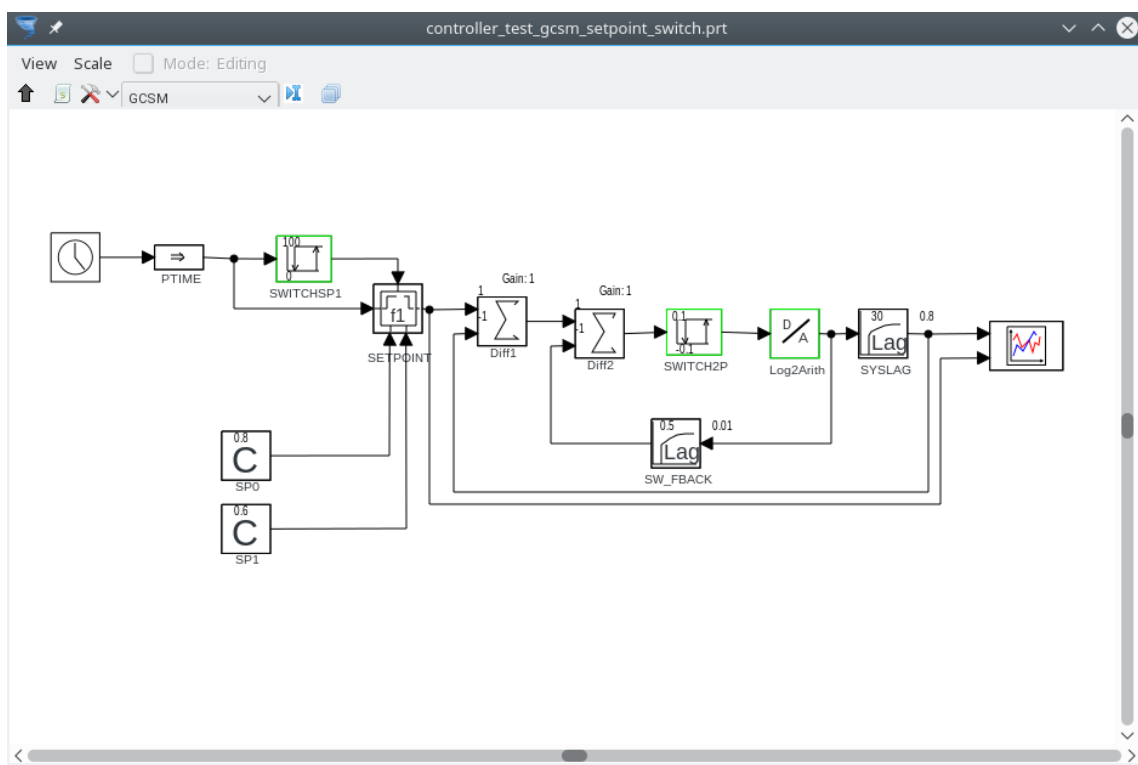


Abb. 4.50 Systemschaltbild einer 2-Punkt-Regelung

Die Ergebnisse wurden mit der Windows Version verglichen und ergeben identische Zeitverläufe (Abb. 4.51).

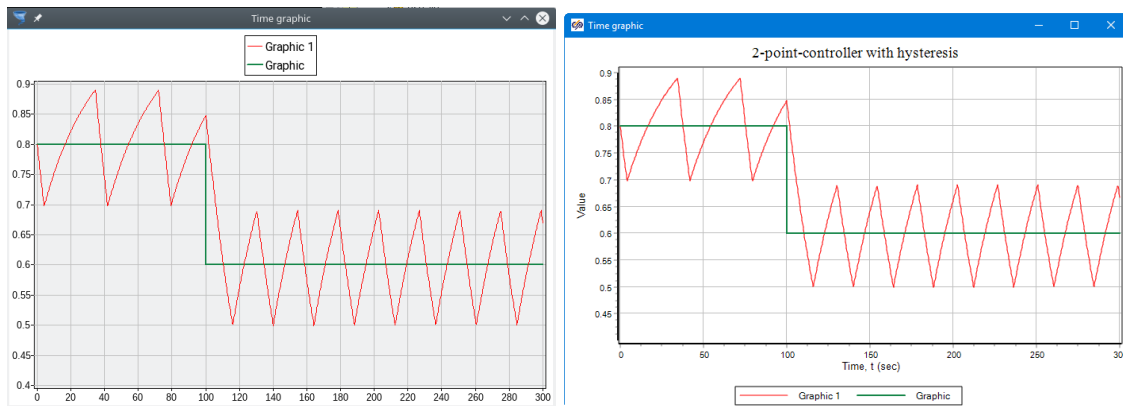


Abb. 4.51 Simulation einer 2-Punkt-Regelung (Linux und Windows)

Da die interaktive Hilfe in AGM auf Microsoft Compiled Help (CHM) basiert war, ist unter Linux ein entsprechender Viewer nötig (z. B. xchm, kchmviewer). Da dieser nicht standardmäßig vorhanden ist, wurde entschieden, die Hilfe auf verlinkte HTML-Seiten umzustellen (siehe Abschnitt 4.1.7), die in jedem WEB-Browser wie z. B. Firefox angezeigt werden können.

4.1.8.2 Import von Systemmodellen aus dem G2-Modellierer in AGM

Neben einem binären Speicherformat unterstützt AGM ein XML-basiertes ASCII-Speicherformat für ein Systemmodell, das für die Übertragung der Systemdaten aus G2 genutzt werden kann. Alle zur Beschreibung eines Modells notwendigen Daten, wie z. B. Systemparameter, Typ, Namen und Verbindungen von Basisblöcken und das grafische Layout, werden in diesem Format in einer Datei (Erweiterung *.xprt) gespeichert. Dieses Datenformat wurde analysiert und Funktionen zum Lesen und Schreiben von Daten in diesem Format entwickelt. Zusätzlich wurde in AGM für alle existierenden Basisblöcke, die in einem Systemmodell vorhanden sein können, eine Template-Datei erzeugt, die die vollständigen XML-Beschreibungen für die Blöcke und Verbindungen enthält.

Da der G2-Modellierer die Systeme in einem proprietären Binärformat speichert und auch kein ASCII-Speicherformat vorhanden ist, musste der G2-Modellierer programmatisch erweitert werden, damit neben den ATHLET-Daten auch die für AGM benötigten Daten der Systemmodelle in lesbarer Form in eine Exportdatei geschrieben werden können. Diese Datei enthält alle wichtigen Daten für sämtliche Signalblöcke mit Namen, Typ, Eigenschaften und den jeweiligen Verbindungen zu anderen Blöcken sowie die Layoutdaten mit den Blockpositionen (siehe Abb. 4.52) und dem Verlauf der Verbindungslinien.

```

59223 >object_41 class_name< LagGRS
59224 >object_41 name< PHIR03
59225 >object_41 visual_props data_0 value< PHIR03
59226 >object_41 custom_props data_0 value<
59227 >object_41 custom_props data_2 value< 1.0
59228 >object_41 custom_props data_2 rexrvalue< 1.0
59229 >object_41 visual_props data_6 value< [ (-565.0 , -195.0) , (-535.0 , -195.0) , (-565.0 , -225.0) , (-565.0 , -161.8) ]
59230 >object_41 visual_props data_19 value< 60.0
59231 >object_41 visual_props data_20 value< 60.0

```

position on workspace

Abb. 4.52 Blockdaten in der Exportdatei des G2-Modellierers

In einem zweiten Schritt werden die in der Exportdatei gespeicherten Informationen in das XML-basierte xprt-Format von AGM übersetzt. Hierzu wurde ein Programm implementiert, das diese Aufgabe übernimmt und dabei auch einige Überprüfungen und Korrekturen vornehmen kann. Diese Korrekturen sind vor allem dadurch notwendig, weil G2 und AGM sich in der Handhabung von Blöcken und Verbindungen unterscheiden. Aufgrund der Fülle an notwendigen zusätzlichen Daten in einer vollständigen AGM-Datei greift das Konvertierungsprogramm auf die Template-Datei zurück, aus der es die notwendige Zusatzinformation für die vollständige Systembeschreibung in AGM beziehen kann. Die Verwendung des Konvertierungsprogramms ist im Anhang B.1 dokumentiert.

Beide Modellierungswerkzeuge können Systemmodelle hierarchisch organisieren. Die Modelle können in verschiedene Arbeitsblätter (Workspaces) unterteilt sein. Die Navigation und Signalübertragung zwischen den Arbeitsblättern erfolgt mittels spezieller Kontrollblöcke. Damit ist eine Abbildung des Systemmodells eins zu eins möglich, das grafische Layout eines Modells (siehe Abb. 4.53 und Abb. 4.54) bleibt bei der Umsetzung also weitgehend erhalten.

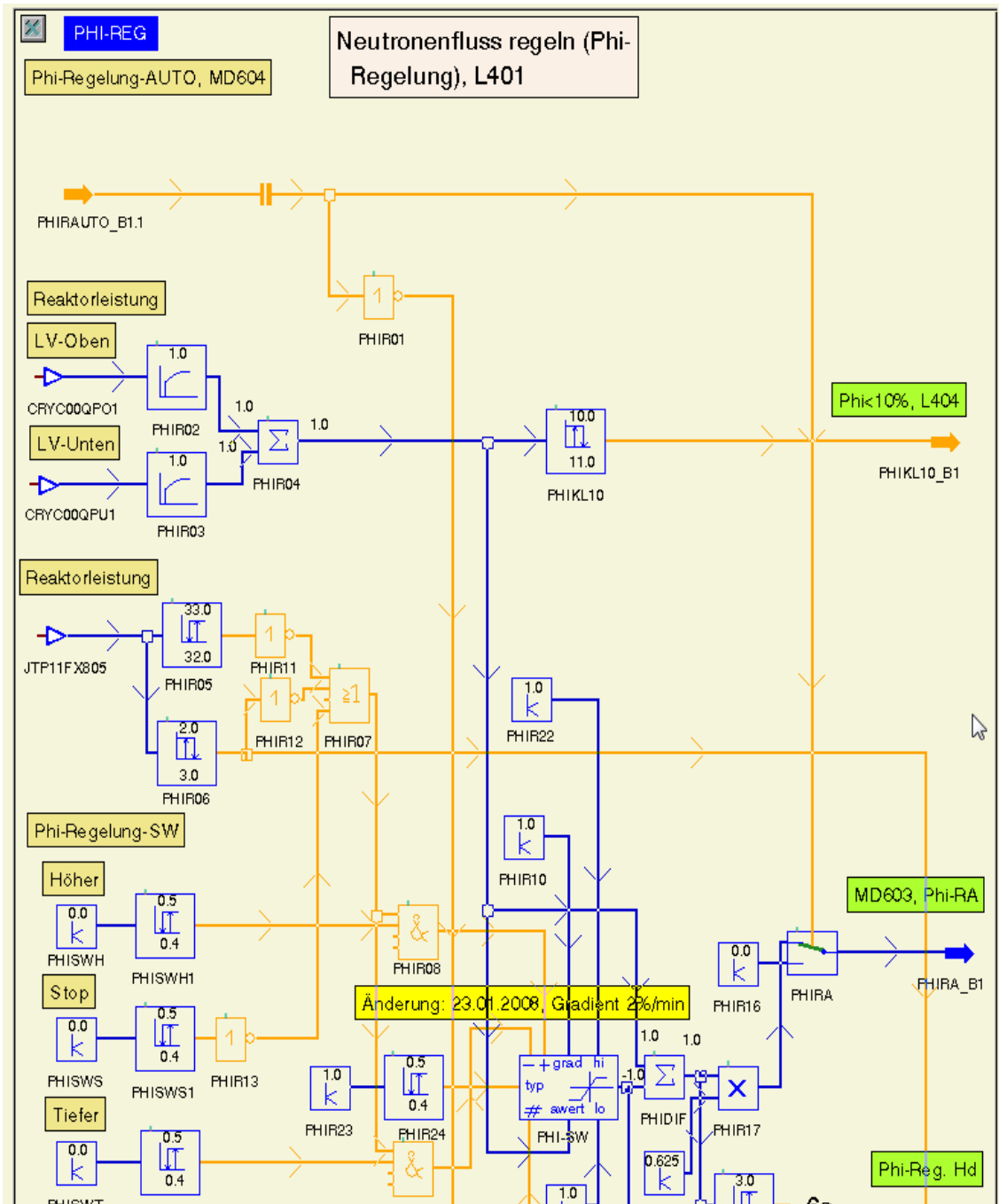


Abb. 4.53 Arbeitsblatt eines Systemmodells im G2-Modellierer

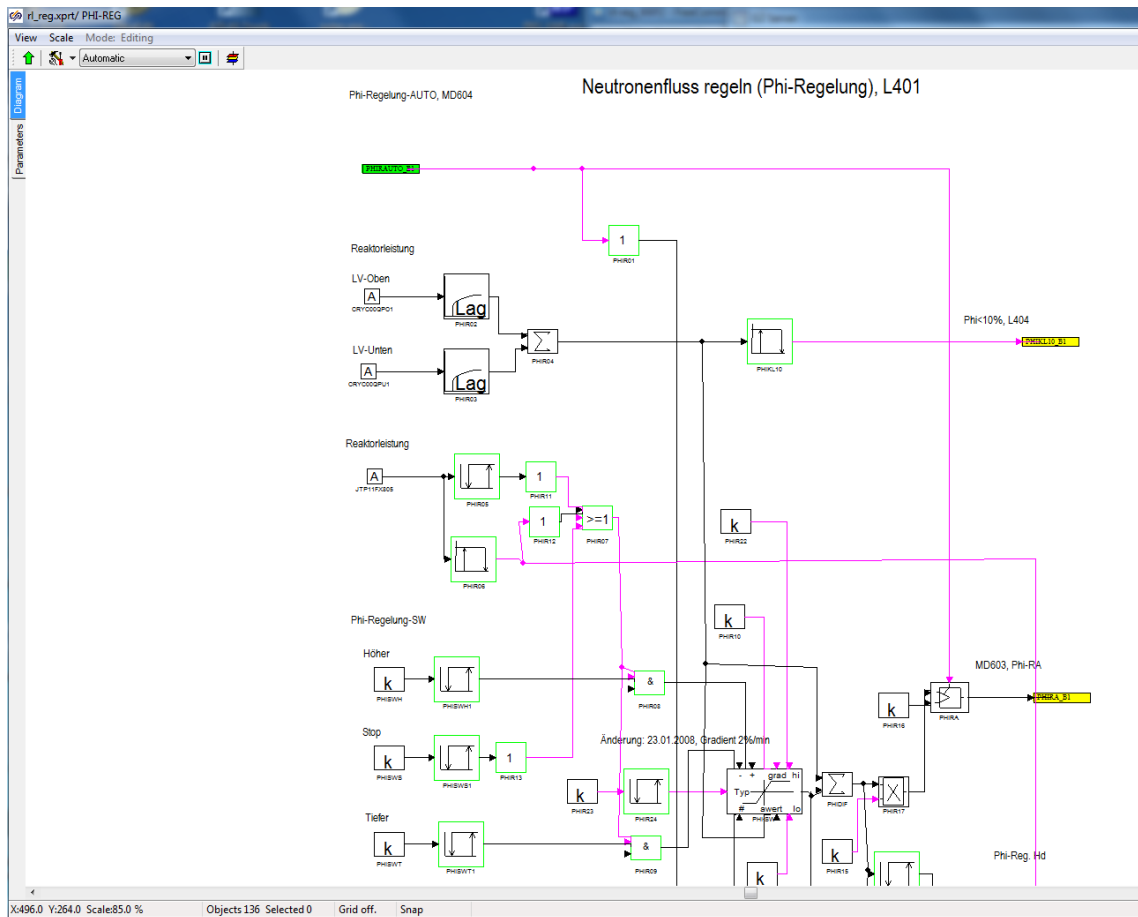


Abb. 4.54 Konvertiertes Arbeitsblatt eines Systemmodells in AGM

Nach einer erfolgreichen Konvertierung eines Systemmodells sollte der Nutzer das importierte Modell in AGM testen. Dazu muss zuerst die Simulation in AGM gestartet werden. Die Simulation überprüft, ob alle Eingänge der Blöcke belegt sind und gibt ggf. weitere Informationen und Warnungen aus. Wenn die Simulation fehlerfrei durchgeführt wird, können die GCSM-Ausgabedaten für ATHLET erzeugt werden.

Die erzeugte ASCII-Datei kann ebenfalls zum Test der Konvertierung verwendet werden, indem die Ausgabedaten von AGM mit den in G2 erzeugten Ausgabedaten verglichen werden. AGM und G2 erzeugen allerdings, wegen unterschiedlicher Algorithmen zur Ermittlung der Blockreihenfolge, verschiedene Reihenfolgen der Daten für die Blöcke in der Ausgabe. Deshalb wurde ein Python-basiertes Programm entwickelt, das die Blöcke (GCSM Signale) alphabetisch sortiert und die Daten in einem einheitlichen Format ausgibt (normiert). Im Anschluss können Unterschiede oder fehlende Daten einfach mit einem Dateivergleichsprogramm („diff-tool“) ermittelt werden. Die Verwendung des Sortierprogramms ist im Anhang B.3 dokumentiert.

Der Konvertierungsworkflow wurde an einem realen Leittechniksystem, der Reaktorleistungsregelung eines DWR, getestet. Die Konvertierung wurde mit dem Werkzeug durchgeführt. Im konvertierten System waren aber manuelle Anpassungen nötig, z. B. beim Layout der Signallinien, bei der Berücksichtigung von Anfangswerten von Signalen und bei Flip/Flop-Blöcken. Mit den neu erzeugten Daten konnte die Simulation in ATHLET durchgeführt werden. Der Vergleich einer Transiente zur Leistungsreduktion ergab aber teils erhebliche Abweichungen in den dynamischen Zeitverläufen. Wegen der vielfältigen Rückwirkungen der Regelung ist es schwierig und zeitaufwendig, die Ursachen detailliert zu untersuchen.

Das Werkzeug zur Transfer der Leittechnikmodelle nach AGM ist in der Praxis einsetzbar. In der Simulation mit ATHLET ergeben sich trotz gleicher Einzelsignale wegen der abweichenden Signalreihenfolge, Unterschiede in den Simulationsergebnissen. Diese müssen bewertet werden und es können Anpassungen in der Modellierung des Systems in AGM nötig sein. Trotz der hohen Automatisierung ist manueller Aufwand erforderlich, der bei sehr komplexen Leittechnikmodellen nicht unerheblich sein kann. Im Einzelfall ist daher abzuwägen, ob der Aufwand für die Modellierung des Systems in AGM gerechtfertigt ist. Dies ist beispielsweise der Fall, wenn das System künftig größeren Änderungen unterworfen ist oder in ähnlicher Form in einem neuen Anlagenmodell benötigt wird.

4.2 Reengineering von ATLAS

4.2.1 Anwendungsfälle von ATLAS

ATLAS wird seit vielen Jahren entwickelt und gegenwärtig in Version 5.1 verwendet. Im Lauf der Zeit ist aus unterschiedlichen Gründen, beispielweise für spezielle Analyseanforderungen, kontinuierlich der Funktionsumfang erweitert worden. Die aktuell in ATLAS vorhandenen Funktionen wurden systematisch erfasst, um ATLAS aus funktionaler Sicht anhand von Anwendungsfällen (use cases) beschreiben zu können. Ziel ist es, diese Anwendungsfälle zu bewerten, z. B. bezüglich der Wichtigkeit, der Einsatzhäufigkeit und spezieller Funktionsdetails, um mit den Ergebnissen die Entwicklungsanforderungen für das Reengineering von ATLAS (ATLASneo) besser planen zu können.

Tab. 4.1 listet die Kurzbezeichnung des Anwendungsfalls, eine nähere Erläuterung der Funktion und eine Bewertung der Wichtigkeit von unwichtig (0) bis sehr wichtig (5) auf. Für einige Standardanwendungen in der Prozessvisualisierung und -steuerung konnte

aufgrund vorliegender Erfahrungen oder Rückmeldungen von ATLAS Anwendern direkt die Wichtigkeit abgeschätzt werden. Für eine Reihe erweiterter Anwendungsfälle in der Visualisierung oder für spezielle Funktionen, wie z. B. die Simulation von Bedienprozeduren, war die Wichtigkeit nicht ausreichend bekannt. Zur Ermittlung der benötigten Bewertung wurde eine computergestützte Umfrage basierend auf Microsoft Sharepoint /WIK 17/ erstellt und an ca. 20 GRS-interne Anwender von ATLAS verteilt. Die Fragen und die Ergebnisse aus der Umfrage sind in Anhang C dargestellt. Aus den Ergebnissen für die Häufigkeit der Verwendung einer Funktion wurde dann eine Wichtigkeit des Anwendungsfalls abgeleitet und in die Tabelle eingetragen.

Tab. 4.1 ATLAS Anwendungsfälle (use cases)

Anwendungsfall	Funktion	Wichtigkeit
Visualisierung		
Keywordtree	Hierarchische Namensliste aller Simulationsvariablen Filter, Suche	5
Wert	Wert einer Variablen alphanumerisch anzeigen	5
Time Chart	Zeitverlauf von einer/mehreren Variablen in einem X/Y-Diagramm darstellen (für beliebige Zeitfenster)	5
Zoom	Bereich, Stufen, Mausrad	5
Gesamtlayout	Alle Bilder/Charts speichern/wiederherstellen	5
Achsendiagramme (X/Y- Charts)		
Layout	Speichern/Wiederherstellen	5
Trendsets	Mehrere Charts (beliebige Anordnung) in einem Fenster	5
Layouteinstellung und -vorgabe	Dialogbasiert, hierarchisch (ähnlich wie „simple property editor“)	4
Zeitachse	Automatische Verlängerung (optional) Nullpunktverschiebung Format (Sekunden oder dd:hh:mm:ss)	4
Y-Achse	Automatische Skalierung oder Min./Max.-Vorgabe	5
Werte	Lineartransformation: $Y = a*y + b$	3
Legendenposition	Freie Position innerhalb und außerhalb des Diagramms	4
Diagramm-Export Vektor	Vektor Formate (SVG, WMF)	4
Diagramm-Export Bitmap	Bitmap Formate (JPG, BMP, TIF)	3
Diagramm-Export Plot	JSPlot APTPlot	2
Diagramm-Export Size	Feste Größe (Pixel, Größenvorgabe)	3
Daten-Export	Wertetabelle einer Variablen für einen Zeitbereich	3

Anwendungsfall	Funktion	Wichtig-keit
Bildelemente und Dynamik		
Wert	Genauigkeit, Format, Phys. Einheit	5
Farbe	Farbänderung entsprechend Grenzwerten oder Farbskala	5
Farbskala	Interaktiv änder- und speicherbar	5
Sichtbarkeit	Bildelemente anzeigen/verbergen	5
Position	Bildelemente verschieben (z. B. für Füllstand)	5
Größe	Bildelemente skalieren (z. B. für Füllstand)	5
Vektor	Vektoren (arrow): Länge, Stärke, Richtung (z. B. für Massenstrom)	5
Bildnavigation	Andere Bilder anzeigen	2
Sounds	Töne / Text abspielen	0
Alarm	Variable überwachen und Messages anzeigen bzw. „sounds“ ausgeben	1
Bitmap	Verwendung einer Bitmap (z. B. Tiff, Jpeg) mit transparenter Farbe	3
X/Y - Diagramm	Beliebiges X/Y Diagramm in einem Bild (X und Y aus Simulationsvariablen) z. B. P-T-Diagramm	2
Dialogelemente	z. B. button, check boxes, dropdown menus, file selection, tooltips, etc.	2
Skripte	Benutzerdefinierte Programme (z. B. Auslösung mehrerer Fehlfunktionen gleichzeitig unter Randbedingungen)	3
Dynamische Variable	Umrechnung mit Funktionen (z. B. Skalierung, Farbwert, ...)	5
Automatisch erzeugte Bilder		
ATHLET Input Grafik	TFO und HCO mit Nodalisierung und lokalen dynamischen Daten	4
ATHLET-CD Bilder	Brenn- Steuerstäbe („Rods“) Debris (Mewa) Schmelze im unteren Plenum (Aida) Lower Head (Lhead)	3
Shapiro-Diagramm (COCOSYS/MELCOR/ASTEC)	H2-Verbrennung („Shapiro“)	3
Prozedurensimulation („operational procedures“, nur für ATHLET)	Logische Verknüpfung von Prozessgrößen („Triggers“) und Aktionen (z. B. Öffnen eines Ventils)	3
ATHLET Charts	Ortsplots von lokalen Größen (z. B. Temperatur) über TFO oder Prioritätskette	4
Steuerung		
Simulatorsteuerung	Freeze (Pause) / Run Steps (forward, backward, any no. of steps) Replay /Jump to time Breakpoint (Stop at time, stop at user defined condition)	4
Snapshot	Restartdaten erzeugen	2

Anwendungsfall	Funktion	Wichtigkeit
Backtrack	Zurückspringen und mit anderen Randbedingungen fortsetzen	2
Fehlfunktion	Variablen ändern (z. B. Auf/Zu, Wert, Rampe)	4
Geschwindigkeitsteuerung	As fast as possible Real time bzw. RT-factor Steps/Frames per second	3
Spezielle Funktionen		
Verteilte Anwendung	Daten / Prozessmodell und ATLAS auf verschiedenen Rechnern	3
Mehrere ATLAS Prozesse	Mehrere ATLAS-Prozesse steuern gleichzeitig für einen Simulationsprozess	2
Manuelle Keywordliste	Namensliste von „Alias“-Namen	0
Aufzeichnung	Screenshot (bel. Bereich) Video	2
Mehrere Datenquellen	Vergleich von verschiedenen Simulationen	4
Interaktive Hilfe	Kontextsensitive Beschreibung der Bedienung	3

4.2.2 ATLASneo

4.2.2.1 Überblick

Die Entwicklung von ATLASneo wurde als Reengineering von ATLAS und Teilerneuerung geplant und begonnen. Die detaillierte Analyse des ATLAS-Bestandscodes zeigte jedoch eine sehr starke Vermischung der Kernfunktionalität und Codeteilen die neu entwickelt werden mussten. Insbesondere die plattformspezifische Implementierung zur Realisierung der grafischen Benutzeroberfläche bereitete hierbei große Probleme. Diese Codeteile wurden sehr eng mit zentralen Routinen verzahnt, was den Zeitaufwand für deren Trennung unverhältnismäßig gesteigert hätte. Hierzu kam der in Fortran entwickelte Bestandscode, welcher zum großen Teil aus Routinen bestand, die vielfach bereits in den Standardbibliotheken von Python /PYT 17/ oder C verfügbar sind, oder welche in diesen Sprachen sehr viel schneller und sicherer reimplementierbar waren. Zudem sollten in der überarbeiteten Version die mittlerweile etablierten Arbeitsvorgänge besser unterstützt und durch die Nutzung standardisierter Dateiformate die Zusammenarbeit mit anderen Anwendungen verbessert werden. Auch die hierzu notwendigen Erweiterungen hätten nur mit viel Aufwand in die alte Codebasis integriert werden können.

4.2.2.2 Benutzeroberfläche

Aus der vorangegangenen Bestandsanalyse folgte der Entschluss für ATLASneo die Benutzeroberfläche von ATLAS unter der Nutzung aktueller Techniken komplett neu zu entwerfen und hierin nach Möglichkeit den Bestandscode als externe Module einzubinden. Wichtig war hierbei die Festlegung der Technologien, die für die Umsetzung der einzelnen Komponenten genutzt werden sollten. Da ATLASneo möglichst wartungsfreundlich und unabhängig vom Betriebssystem implementiert werden sollte, lag der Fokus hierbei auf der Verfügbarkeit, Modularität und leichten Erweiterbarkeit.

Als weitere Anforderungen wurden an die neu zu entwickelnde Benutzeroberfläche die Verbesserung der Bedienbarkeit und die Angleichung an allgemeine Bedienstandards festgelegt. Für die Realisierung wurde Qt /QTC 17/ als zu Grunde liegende GUI-Bibliothek ausgewählt. Da Qt frei und plattformunabhängig verfügbar ist, alle Konzepte moderner Benutzeroberflächen unterstützt und in der Softwarewelt sehr verbreitete Anwendung findet, kann hier von einer stabilen und zukunftssicheren Grundlage für die weitere Entwicklung ausgegangen werden.

4.2.2.3 Datenformate

Im Workflow der Anwender nehmen ATLAS, und somit zukünftig auch ATLASneo, eine besondere Rolle ein. Als zentrale Anwendung zur Analyse, Auswertung und Aufbereitung von Simulationsdaten und zur Simulationssteuerung laufen in ihr sämtliche von anderen Programmen¹ erzeugten Daten zusammen. So müssen für eine detaillierte Visualisierung oder eine Online-Simulation die Ausgabedateien von einigen anderen, von ATLAS meist unabhängig entwickelten Programmen verarbeitet werden können. Um diesen Datenaustausch zu ermöglichen, wurde in der Vergangenheit eine Reihe eigener Datenformate definiert und implementiert. Zum Zeitpunkt dieser Entwicklungen existierten die heute verfügbaren Standardformate oft noch nicht, was auch die Entwicklung von Tools zur Erzeugung und Verarbeitung dieser Datenformate notwendig machte. Hieraus entstand eine ganze Anzahl von Programmen rund um ATLAS und ATHLET, welche deren Zusammenarbeit und den Datenaustausch untereinander erst möglich machten.

¹ hauptsächlich aber nicht ausschließlich ATHLET, ATHLET-CD, AIG, APG

In ATLAS werden zweidimensionale Bilder der Modellgeometrie, von Steuerelementen zur dynamischen Darstellung von Simulationsdaten und zur Interaktion mit den Modellkomponenten im GRS-eigenen Vektorgrafikformat "APG" gespeichert und mit speziell dafür entwickelten Programmen (AIG: ATHLET Input Grafic, APG: ATLAS Picture Generator) teilautomatisch oder auch manuell erstellt.

Um sich auf die Weiterentwicklung von Anwendungen und deren Anpassung auf neue Bedürfnisse konzentrieren zu können, ist es wichtig den Aufwand für Wartung und Codepflege so gering wie möglich zu halten. Hierzu kann die Verwendung von Standardformaten einen großen Beitrag leisten, da die Funktionalitäten zum Lesen und Schreiben dieser Datenformate praktisch immer als freie Bibliotheken zur Verfügung stehen². Diese können in der Regel mit geringem Aufwand in eigene Anwendungen eingebunden werden und sorgen somit für Datenkompatibilität mit etablierten Anwendungen.

Natürlich soll ATLASneo die in ATLAS vorhandenen Visualisierungsmöglichkeiten anbieten und nach Möglichkeit um bisher fehlende Darstellungsformen erweitern. Eine flexible Visualisierung von Simulationsgrößen benötigt eine geometrische Beschreibung, welche es erlaubt die Daten im Kontext anderer Informationen, z. B. als schematische Schaltbilder darzustellen. Da diese Beschreibung völlig vom Fall der gerade untersuchten Simulation abhängt, muss diese möglichst einfach vom Anwender erstellt und auf die jeweiligen Anforderungen angepasst werden können. Das dafür bisher verwendete APG-Format sollte deshalb durch ein Standardformat ersetzt werden, welches die benötigten Anforderungen zur geometrischen Beschreibung von Visualisierungsszenen erfüllt und gleichzeitig durch hohe Verfügbarkeit leicht integriert werden konnte.

Durch seine Möglichkeiten, Flexibilität und Verbreitung stellte sich das SVG-Format (Scalable Vector Graphics) /SCA 11/ als eines der dafür am besten geeigneten Optionen heraus und wurde als in ATLASneo verwendetes Grafikformat festgelegt.

Der heutzutage etablierte Web-Grafikstandard SVG weist viele Überschneidungen mit den Anforderungen auf, die APG abdeckt. SVG ist ein XML-Format, das von modernen Browsern erkannt und dargestellt wird. Alle wichtigen APG Grafikprimitive und Transformationen sind enthalten. Die für ATLAS wichtige Dynamisierung der Bilder ist über die Manipulation des "Document Object Model" (DOM), der browserinternen Repräsentation

² Sollte das nicht der Fall sein, kann eigentlich nicht von einem Standardformat gesprochen werden.

der verschachtelten XML-Struktur, möglich. Ein SVG-Element kann mit Attributen versehen werden, die neben der Konfiguration das individuelle oder objektbasierte Ansprechen einzelner Bestandteile eines Bildes ermöglichen. Nicht im Standard enthaltene Attribute sind möglich und lassen sich für Erweiterungen nutzen. Die hierdurch visualisierten Simulationsdaten können anschließend durch die im Browser eingebettete JavaScript-Laufzeitumgebung eingespielt werden und ermöglichen so eine schnelle Aktualisierung der dargestellten Szene.

Durch eine Migration von APG zu SVG /SCA 11/ bietet sich die Möglichkeit auf bereits existierende Softwarelösungen für die Editierung (z. B. mit Inkscape /INK 17/) und Darstellung (z. B. WebKit /WEB 91/, AIG oder Blink /BLI 17/) von Vektorgrafik zurückzugreifen und auch von zukünftigen Verbesserungen auf diesem Gebiet zu profitieren. Durch die sehr hohe Verfügbarkeit und der damit zusammenhängenden Technologien erscheint die Nachhaltigkeit dieses Ansatzes gesichert.

Damit auch zukünftig die in der Vergangenheit erstellten grafischen Reaktorbeschreibungen für Simulatoren genutzt werden können, benötigt man für den Übergang von APG zu SVG Konvertierungswerkzeuge. Zu diesem Zweck wurde das im Abschnitt 4.2.3 beschriebene Werkzeug „apg2svg“ entwickelt.

4.2.2.4 Fazit

Aus der Analyse der bestehenden ATLAS-Version ging eine Vielzahl an Anforderungen an ATLASneo hervor, welche nicht nur neue oder verbesserte Funktionalität, sondern vor allem auch die Lösung softwaretechnischer Probleme umfasste. Für einen neu entwickelten Code sollten natürlich die aktuellen Softwarestandards angelegt werden, um so zumindest für die neuen Anwendungsteile zukünftig von einer verbesserten Wartbarkeit und einer erleichterten Weiterentwicklung profitieren zu können. Hieraus ergibt sich aber mittelfristig eine Mischung von alten und neuen Codeteilen, welche zusammenarbeiten sollen.

Zur Realisierung des Projektes musste deshalb ein Konzept gefunden werden, welches erlaubt flexibel mit dem bestehenden Code umzugehen und diesen einbinden zu können, ohne sich dafür bei der Nutzung neuer Technologien einschränken zu müssen.

Einen sehr wichtigen Beitrag leistete hierzu die Festlegung darauf, den zentralen Anwendungsrahmen in der Programmiersprache Python /PYT 17/ zu implementieren.

Python bringt zum einen die Flexibilität von Skriptsprachen mit, erlaubt es aber nativ implementierte Codes (als shared library vorliegend) sehr effizient zu nutzen und diese zur Zusammenarbeit (z. B. Kopplung) zu bewegen. Diese Fähigkeit wurde bereits in einer schier unüberschaubaren Anzahl von Open Source Projekten genutzt, wodurch die mittlerweile riesige in Python zur Verfügung stehende Funktionsvielfalt entstehen konnte. Darum bietet sich hier auch die Gelegenheit, dieses Potential für die zukünftige Anwendung und Weiterentwicklung von ATLASneo zu erschließen und bereits implementierte „Insellösungen“ in den Anwendungsrahmen nach und nach zu integrieren.

Somit wird ATLASneo in Python /PYT 17/ und Javascript/ECMAScript /WIK 17a/ entwickelt. Als Fensterumgebung dient die C++-Bibliothek Qt_/QTC 17/, die über das Paket PySide /PYS 15/ in Python ansprechbar ist. Qt bietet mit der Klasse QWebView /QWE 17/ eine Schnittstelle zur HTML-Rendering-Engine WebKit /WEB 91/, die Javascript-Code interpretiert. Des Weiteren kommt für die Darstellung von Graphen PyQt-Graph /PYQ 17/ zum Einsatz.

4.2.3 Erzielte Ergebnisse

Die Entwicklung von ATLASneo wurde gemäß den vorab festgelegten Schwerpunkten vorangetrieben. Im Laufe der Arbeiten musste deren Priorisierung natürlich entsprechend der entwicklungstechnischen Bedürfnisse angepasst werden, wodurch sich z. B. die bereits begründeten Veränderungen in der Vorgehensweise zum Reengineering des ATLAS-Bestandscodes ergaben. Die folgenden Abschnitte beschreiben den aktuellen Stand sowie die bisherigen Teilergebnisse im Einzelnen.

4.2.3.1 Konvertierung von Bildern mit apg2svg

Um bestehende ATLAS-Bilder im APG-Format im ATLASneo-Bildmodul "EIDOS" verwenden zu können und eine Referenzimplementation für den Aufbau der SVG-Bilddateien zu erstellen, wurde der Konvertierer "apg2svg" geschrieben. Er ist in der Sprache Nim /NIM 17/ programmiert. Das Programm kann die häufigsten, statische Bildeigenschaften betreffende APG-Befehle (in der Auflistung in Klammern) in das SVG-Format übersetzen:

- Punkte (P)
- einfache oder parallele (PL) Linienzüge (S, R, L, U, D) und daraus zusammengesetzte Polygone

- Vektoren (V)
- Splines (SPLINE)
- Rechtecke (BOX)
- Kreise (CR), Ellipsen und Abschnitte davon (PC, PE)
- Text (M)
- ausgefüllte Flächen (ARFILL, ENDFILL)
- Winkel (A) | Bitmap-Bilder (IMAGE, SETIMAGE)
- Transformationen (TRANSF)
- APG-Makros (VALVE, PUMP, PRHT, LAG, LIM, SBKFL)
- Objekte und Prioritäten (OPOBJ, CLOBJ)
- Speichern (#), Abrufen (B) und Verschieben (S#, R#, L#, U#, D#) von Punkten
- Variablendefinition und Auswertung (USDATA)
- IDs für Objekt-Rechenknoten (DEFATH, PICKVAR, ENDPICK)
- Angabe der Visualisierungsreihenfolge, Prioritäten
- Farben (GSPLCI)
- Linienattribute (GSLWSC, GSLN, GSPLTR, LALIAS)
- Vektorattribute (VECTATT)
- Flächenattribute (GSFACI, EDGE, GSFAIS, GSFASI, GSFATR)
- Kommentare (!)

Die feste Definition von Farbskalen in der Bilddatei ist nicht mehr nötig, weil diese in dem neuen Visualisierungsmodul während des Programmlaufs datengesteuert erzeugt werden. Deswegen entfällt die Interpretation der APG-Befehle CSCALE, COLSC, COLIV und SETCOLS. Bei der Umsetzung wurde darauf geachtet, durch die Definition von wiederverwendbaren SVG-Symbolen möglichst kompakte Dateien zu erzeugen, was möglicherweise auch Vorteile bei der Darstellung bringt. Grafische Bestandteile von ATHLET-Objekten und -Knoten werden in benannten und mit Klassenattributen versehenen SVG-Gruppen zusammengefasst.

Da die Steuerung der Darstellung von zeitabhängigen Simulationsvorgängen in Zukunft weitestgehend von den Bilddateien entkoppelt sein soll, werden die die Dynamik betreffenden APG-Befehle bei der Übersetzung ignoriert. Zur Umsetzung der Animations- und Interaktionsmöglichkeiten, die APG zur Verfügung stellt, kommen in EIDOS für Standardaufgaben wie z. B. Farbanimation, dynamisierte Texte oder Füllstände vorgefertigte, JavaScript basierte Lösungen zum Einsatz. Weitergehende Anforderungen sollen in Zukunft von einer Programmbibliothek abgedeckt werden.

Im Folgenden werden einige Anwendungsbeispiele gezeigt. Dabei befindet sich links immer die Darstellung einer APG-Datei und rechts das SVG-Bild als Ergebnis der Konvertierung. Weil die Übersetzung allein auf den APG-Befehlen aufbaut sind die Ergebnisse was die Seitenverhältnisse betrifft, nicht immer korrekt, da bestimmte Transformationen im alten ATLAS-Programm erst beim Rendering durchgeführt werden. Außerdem erzeugt die ATHLET Input Grafik zusammenhängende Flächen oft aus mehreren separaten Objekten, so dass im konvertierten SVG-Bild innerhalb eines Kontrollvolumens störende Linien aus den Flächenumrissen entstehen. Es ist geplant, diese Teilvolumina mithilfe einer externen Bibliothek (Lib2Geom, <https://gitlab.com/inkscape/lib2geom>) aus dem Inkscape-Projekt miteinander zu verschmelzen.

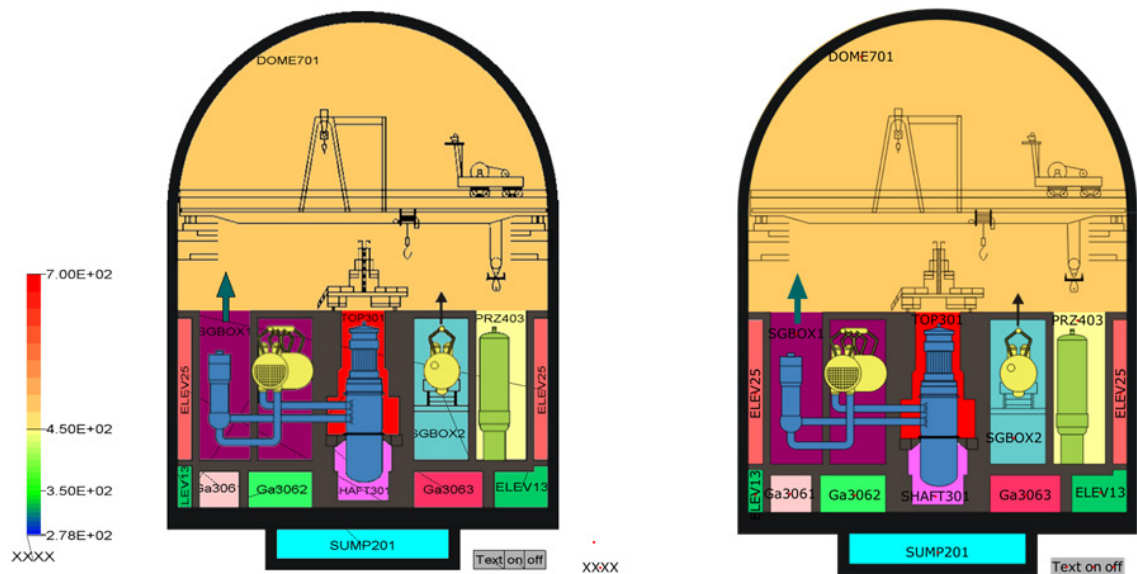


Abb. 4.55 Anwendungsbeispiel „Containment“



Abb. 4.56 Linien und Flächeneigenschaften

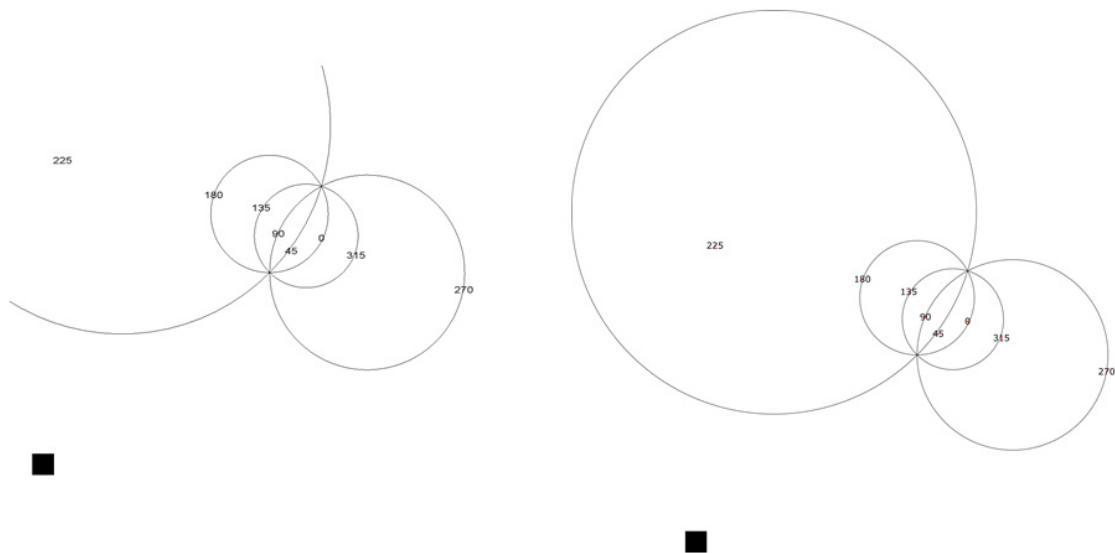


Abb. 4.57 Einstellwinkel und Kreise

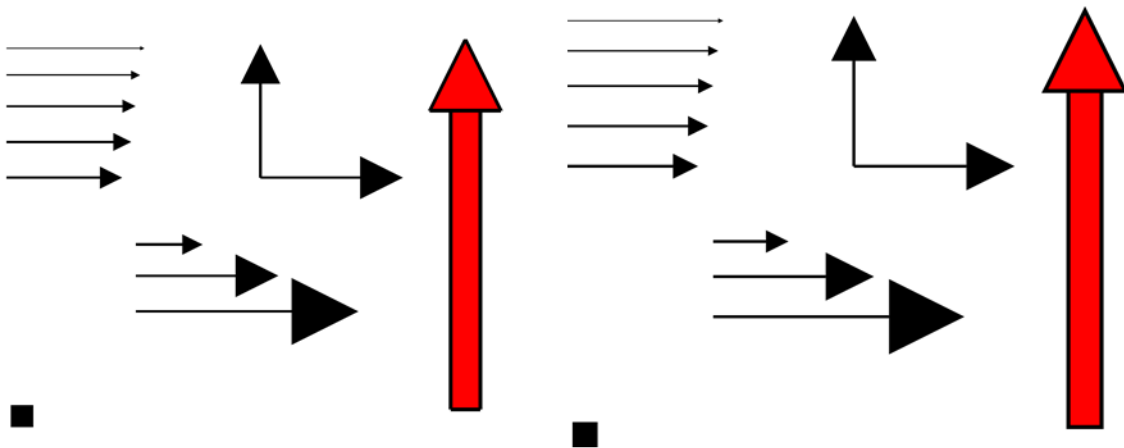


Abb. 4.58 Vektoren

4.2.3.2 Der Applikationsrahmen

Für die Neuimplementierung der Benutzeroberfläche von ATLASneo wurde vorgesehen, die Funktionalität zur Visualisierung und Simulationssteuerung in Form von eigenständigen, dockbaren Fenstern (DockWidgets) zu implementieren, welche eine sehr hohe Flexibilität bieten und den Benutzern erlauben, das Layout der Oberfläche individuell an die verschiedenen Bedürfnisse bzw. Arbeitsweisen anzupassen. Der Applikationsrahmen dient dazu, die einzelnen Funktionseinheiten (Funktionsmodule) zu organisieren und den Benutzern eine einheitliche Bedienoberfläche anzubieten.

Der Rahmen enthält eine Sitzungsverwaltung, welche den aktuellen Zustand der Anwendung in Form einer Sitzungsdatei speichern und wiederherstellen kann. In einer Sitzung

werden dabei sowohl die Gesamtheit der aktuell geöffneten Funktionsmodule als auch deren Anordnung innerhalb des Rahmens gespeichert. Durch diese Funktion können für verschiedene Aufgabenstellungen, wie eine Online-Simulation oder bestimmten Auswertungsarten, jeweils eine geeignete Auswahl und Anordnung benötigter Funktionsmodule vorbereitet werden. Diese können dann von den Anwendern genutzt werden, um ATLASneo direkt in einem bestimmten Betriebsmodus zu starten. Das Speichern der Sitzungsdatei geschieht beim Schließen der Anwendung automatisch, kann aber auch gezielt über das Hauptmenü erfolgen.

Eine weitere Aufgabe dieses Anwendungsteils ist es, die beim Start automatisch erkannten Module in einer Werkzeugleiste darzustellen und diese bei einem Klick auf das entsprechende Icon zu starten (vgl. Abb. 4.59). Wie in der Abbildung dargestellt, kann auf das Icon eines Funktionsmoduls mehrfach geklickt werden, was dazu führt, dass entsprechend viele Instanzen dieses Moduls gestartet werden. Um die einzelnen Modulinstanzen besser unterscheiden zu können, wurde die Benennung und der angezeigte Titel auf ein einheitliches Schema angepasst. Die automatische Benennung gestarteter Funktionsmodule stellt sicher, dass diese einfach identifiziert werden können. Der Titel der Module kann gemäß ihrer Funktion und ihres aktuellen Inhalts festgelegt werden. Alle geöffneten Funktionsmodule werden auch in der Hauptmenüleiste, im Menüpunkt „View“ eingetragen, und können hierüber gezielt in den Vordergrund geholt werden.

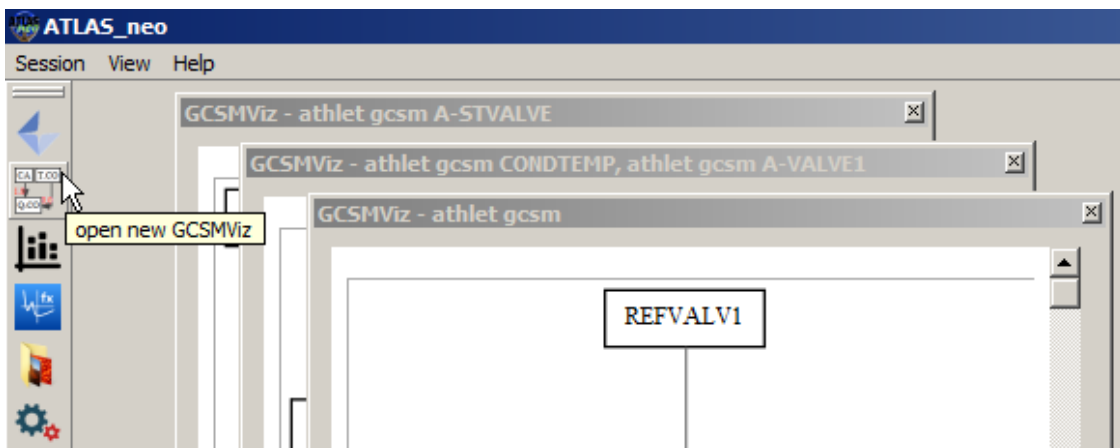


Abb. 4.59 Mehrfachstart und Titel von Funktionsmodulen

4.2.3.3 Modularer Aufbau in ATLASneo

Alle Funktionsmodule in ATLASneo werden zwar vom Anwendungsrahmen erkannt, gestartet und verwaltet, allerdings hat dieser keine genauere Kenntnis über deren Inhalt und Funktionsweise. Diese strenge Modularität stellt sicher, dass neue Funktionen

voneinander unabhängig entwickelt werden können und sich diese ohne Veränderungen am Rahmen oder in anderen Teilmodulen in die Gesamtanwendung integrieren. Das bedeutet auch, dass alle Funktionen optional nutzbar sind und bei Bedarf sehr einfach deaktiviert oder ausgetauscht werden können. Hiermit wird die Grundlage dafür gelegt, dass die Anwendung auch zukünftig leicht und flexibel weiterentwickelt und an neue Bedürfnisse angepasst werden kann.

Eine Konsequenz der Modularität in ATLASneo ist, dass auch die Funktionsmodule grundsätzlich nichts voneinander wissen. Das bedeutet z. B., dass die Implementierung von Darstellungsmodulen, wie das Plotmodul (siehe Abschnitt 4.2.3.8), keine Annahmen über den Aufbau oder Inhalt anderer Module macht, sondern sich ganz auf die Verarbeitung der ihm zugeteilten Daten konzentriert. Dadurch spielt es für ein Modul keine Rolle woher die Daten kommen, die es verarbeiten soll, solange diese in einem von ihm unterstützten Format vorliegen. So wird eine sehr hohe Flexibilität und Wiederverwendbarkeit von Modulen erreicht.

Python-Packages als Einheiten für Funktionsmodule

Um die Trennung zwischen Anwendungsrahmen und den Funktionsmodulen umzusetzen, war es naheliegend, ATLASneo durch Aufteilung in Python-Packages zu strukturieren. Dieser Ansatz wurde gewählt, um die einzelnen Module möglichst eigenständig zu halten und so die Gesamtanwendung ohne spezifische Konfigurationen, wie z. B. Umgebungsvariablen und Suchpfade, ablauffähig zu halten. Natürlich kommt es vor, dass einzelne Funktionsmodule spezielle externe Pakete oder Ressourcen verwenden, welche dann zur Verfügung gestellt werden müssen. Durch die Struktur von Python-Packages können entsprechende Vorkehrungen aber innerhalb des Modulpakets getroffen und z. B. im Initialisierungsteil des Packages (`__init__.py`) umgesetzt werden. Somit beeinflussen diese nicht die Abauffähigkeit anderer Funktionsmodule oder des Anwendungsrahmens. Durch diese Aufteilung wird es auch möglich, die Anwendungsteile hierarchisch zu organisieren und strikt voneinander getrennt zu realisieren.

Darüber hinaus erlaubt dieser Ansatz eine flexible Einbindung von neu entwickelten Erweiterungen und Funktionsmodulen. Vorhandene Erweiterungen, die als Python-Packages vorliegen, werden beim Start des Anwendungsrahmens automatisch erkannt und entsprechend ihres Typs in die dafür vorgesehenen Kontrollelemente der Benutzeroberfläche, wie die Werkzeugleiste oder die Auswahlmenüs (siehe Abb. 4.59) integriert.

Hierdurch reduziert sich der Wartungsaufwand für die zukünftige Funktionsentwicklung erheblich.

4.2.3.4 Drag-and-Drop

Bei aller Modularität in Aufbau und Entwicklung, müssen die Module trotzdem miteinander kommunizieren können, um zur Laufzeit Daten auszutauschen und Aktionen auszulösen. Dieser Widerspruch wird dadurch aufgelöst, indem der Anwendungsrahmen einen Standardmechanismus zur Kommunikation bereitstellt den die Funktionsmodule verwenden, um im Kontext der Gesamtanwendung ATLASneo zu arbeiten. Das hierfür vorbereitete Konzept sieht vor, dass Kommunikationsverbindungen zwischen Modulen erst vom Benutzer hergestellt werden. Funktionsmodule, welche z. B. aus Dateien eingelebene Daten darstellen und ggf. auch bearbeiten, können für andere Module als Datenquelle agieren und mit diesen interaktiv verknüpft werden. Dem Anwender muss hierfür ein einfaches Bedienungskonzept bereitgestellt werden, um diese Verknüpfungen einfach und intuitiv vornehmen zu können. Mittlerweile hat sich für diese Art von Benutzerinteraktion in praktisch allen oberflächenbasierten Anwendungen die Methode des „Drag-and-Drop“ etabliert, mit der ein Anwender ein Objekt mit der Maus wählen, ziehen (drag) und in einem für die Verarbeitung geeigneten Zielbereich durch das Loslassen des Mausbuttons fallenlassen (drop) kann. Diese Aktionen sind beliebig wiederholbar und eignen sich grundsätzlich dazu, beliebige Quell- und Zielbereiche miteinander in Verbindung zu bringen oder diese Verbindungen auch wieder zu lösen. Hierdurch wird deutlich, welche zentrale Rolle der „Drag-and-Drop“-Mechanismus für die Bedienung von ATLASneo spielt, und warum es wichtig ist, diesen flexibel und generisch zu implementieren, um letztlich die Funktionsmodule zu verknüpfen und zusammenarbeiten zu lassen.

Konzeption

Um die für dieses Bedienkonzept notwendige Flexibilität zu erreichen wurde der initial implementierte „Drag-and-Drop“-Mechanismus grundlegend überarbeitet und über ein generisches, MIME³-type-basiertes Protokoll implementiert. Hierdurch ist es grundsätzlich möglich, alle Arten von Daten und Datenstrukturen zwischen den Funktionsmodulen

³ Der **MIME**-Type (Multipurpose Internet Mail Extension) oder auch **Internet Media Type** dient allgemein zur Klassifizierung von versendeten Daten.

hin- und herziehen. Wichtig ist, dass dabei Metadaten mit ausgetauscht werden, welche die zu verarbeitenden Daten klassifizieren und dem Zielmodul, in das ein Datenobjekt gezogen wurde, alles Wichtige über ihre Art und Struktur mitteilen. Anhand dieser Metadaten kann es entscheiden, ob es diese Art von Daten verarbeiten kann oder nicht. Nur falls das Zielmodul für diesen Datentyp vorbereitet ist, wird ein "Drop", (d. h. Loslassen des Mausbuttons) akzeptiert und die Daten können verarbeitet werden. Andernfalls werden die Aktion und damit auch die Daten als ungültig eingestuft und ignoriert.



Abb. 4.60 Drag-and-Drop in Aktion: der Maus-Zeiger signalisiert durch seine Form, ob eine Verknüpfung möglich ist (links) oder nicht (rechts)

Wie in Abb. 4.60 gezeigt, verändert sich während des Vorganges der Maus-Zeiger und signalisiert so dem Benutzer, ob ein Zielbereich die gezogenen Daten akzeptiert oder diese ablehnt. Hieraus ergibt sich eine intuitive Bedienung, welche auch verhindert, dass zueinander inkompatible Funktionsmodule miteinander interagieren.

Implementierung von Drag-and-Drop

Wie oben beschrieben, sieht das implementierte Drag-and-Drop-Protokoll vor, dass Funktionsmodule, welche das Ziehen (drag) von Datenobjekten erlauben, diese Objekte durch Angabe eines MIME-type-Strings klassifizieren. Um das Ablegen (drop) von Daten im Zielbereich zu kontrollieren, kann jedes Funktionsmodul selber entscheiden, ob und welche Drop-Daten es akzeptiert und wie diese verarbeitet werden. Im einfachsten Fall ist dieses eine „Single-Shot“-Verarbeitung, d. h. die Daten werden nur einmalig verarbeitet und z. B. in ein Eingabefeld o. ä. eingefügt. Eine weitere und vielfach genutzte Möglichkeit ist das Übertragen von Daten-Referenzen, welche dem Zielmodul ermöglichen eine direkte Verbindung zum Ursprung der benötigten Daten aufzubauen und diese je nach Bedarf auch wiederholt auszulesen und den jeweils aktuellen Stand neu verarbeiten zu können. Besonders in diesem Fall ist es wichtig, Zielmodule über eine Aktualisierung der referenzierten Daten benachrichtigen zu können. So wird vermieden, dass diese ständig selbst auf eine Veränderung hin prüfen müssen oder unnötige Neuverarbeitungen der Daten durchführen.

Um auf diese Anforderungen reagieren zu können, wurde das Drag-and-Drop-Protokoll auf die Übertragung von speziellen Datenobjekten vorbereitet. Abgesehen von allen Standard-MIME-types, wie Text und Bildformate, welche auch durch Qt /QTC 17/ genutzt werden, gibt es deshalb in ATLASneo aktuell folgende zusätzlichen Typen:

Tab. 4.2 Erweiterte MIME-types, welche bisher in ATLASneo verwendet werden

MIME-type	Bedeutung
application/url-<type>-lists	2er-Tupel aus Listen mit URL-Strings und Daten-Referenzen vom Type <type>.
application/src-model	Das Quell-Modelobjekt, das den Zugriff auf die referenzierten Daten verwaltet.
signal/dragDataChanged	Das Qt-Signal, welches die Veränderung von referenzierten Daten signalisiert.
slot/dropDataChanged	Eine Slot-Funktion über die das DropWidget der Quelle Datenänderungen signalisieren kann.
slot/<name>	Eine Slot-Funktion, welche auf ein Signal <name> reagieren möchte

Funktionsmodule, welche das "Drophen" von Datenobjekten erlauben, können durch eine Liste von MIME-type-Strings festlegen, welche Art von Daten sie verarbeiten können. Das Protokoll reagiert darauf automatisch mit der entsprechenden Steuerung der „Dropability“ und leitet nur unterstützte Datentypen an das Funktionsmodul weiter. Abgesehen vom rein typbasierten Test, kann der Zielbereich (DropWidget) auch eine detailliertere Überprüfung der gezogenen Daten durchführen, um zu entscheiden ob, und wenn ja, welcher Teil der so zugewiesenen Daten verarbeitet werden kann. Für die abgelehnten Datenelemente kann das DropWidget optional Meldungen generieren, welche dann in Form eines Tooltips angezeigt werden (siehe Abb. 4.61). Dadurch kann dem Benutzer auf einfache Weise mitgeteilt werden, warum die von ihm vorgesehenen Daten nicht verarbeitet werden können. Dadurch kann die intuitive Bedienbarkeit der Anwendung deutlich gesteigert werden.

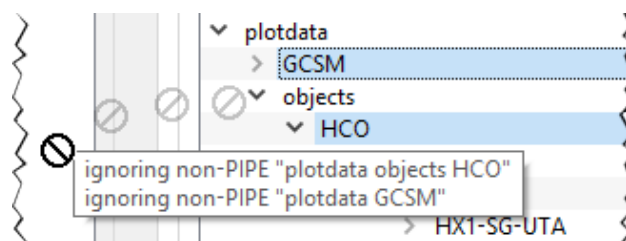


Abb. 4.61 Drag-and-Drop ungeeigneter Daten: Der Tooltip am Maus-Zeiger zeigt dem Benutzer welche Datenelemente ignoriert werden und warum.

4.2.3.5 HDF5-basierte Verarbeitung von Simulationsdaten

Eine der wichtigsten Funktionen in ATLASneo ist die Verarbeitung vieler unterschiedlicher Simulationswerte und deren zeitliche Entwicklung. Diese Anforderung gilt sowohl für online laufende als auch für bereits abgeschlossene Simulationen. Darüber hinaus muss eine zeitliche Navigation entlang der zur Verfügung stehenden Daten möglich sein. Um diese Anforderungen realisieren zu können ist es sehr wichtig die dabei zu erwartenden Datenmengen handhaben zu können. Naive Techniken, wie ein einfaches Ablegen im Arbeitsspeicher oder die Auslagerung in einfachen Dateien würden zur Laufzeit sicher viele Grenzen überschreiten, wodurch ATLASneo für Simulationen mit großem Datenaufkommen nicht mehr einsetzbar wäre.

Aus diesem Grund wurde für die Speicherung und Pufferung von Daten auf das Speicherformat HDF5 /HDF 17/ umgestellt. HDF5 ist ein offenes, auf große Datenmengen ausgelegtes Speicherformat und unterstützt sowohl die effiziente Speicherung der Daten als Datei, als auch den optimierten Lesezugriff und das Caching von Datenbereichen. Eine weitere sehr wichtige Eigenschaft des Formats ist, dass HDF5 keine feste Struktur vorgibt, sondern die Daten flexibel nach einem hierarchischen Schema als voneinander unabhängige Tabellen abgelegt werden können. Da die Ablagestruktur auch Einfluss auf die Zugriffsgeschwindigkeit hat, war es zudem wichtig eine Struktur festzulegen, welche die Daten für den Anwender übersichtlich organisiert und dennoch die Anforderungen bei den zu erwartenden Datenauswertungen möglichst gut berücksichtigt.

Zur Erstellung von HDF5-Dateien direkt in ATLASneo wurde ein Modul (HDF5Writer) implementiert, welches die zeitliche Entwicklung von Werten protokollieren und als HDF5-Struktur ablegen kann. Das Modul aktualisiert die intern verwalteten Puffer laufend mit den neuesten Daten und erlaubt somit jederzeit den Zugriff auf die kompletten Zeitreihen der überwachten Speicherbereiche. Dadurch bildet die bereitgestellte HDF5-Struktur die Datengrundlage für sämtliche Funktionsmodule die mit Zeitreihen arbeiten.

4.2.3.6 Das HDF5Panel

Das Anwendungsspektrum von ATLASneo umfasst neben online durchgeführten Simulationen auch die Auswertung von Ergebnissen abgeschlossener Simulationsläufe. Hierfür müssen auch anderweitig erzeugte Ergebnisdateien eingeladen werden können. Zu diesem Zweck wurde das in Abb. 4.62 gezeigte Funktionsmodul (HDF5Panel) implementiert, welches HDF5-Dateien laden und deren Daten als Baum anzeigen kann. Das

HDF5Panel stellt die Daten über den Drag-and-Drop-Mechanismus zur Verfügung und kann somit als Datenquelle für andere Funktionsmodule verwendet werden.

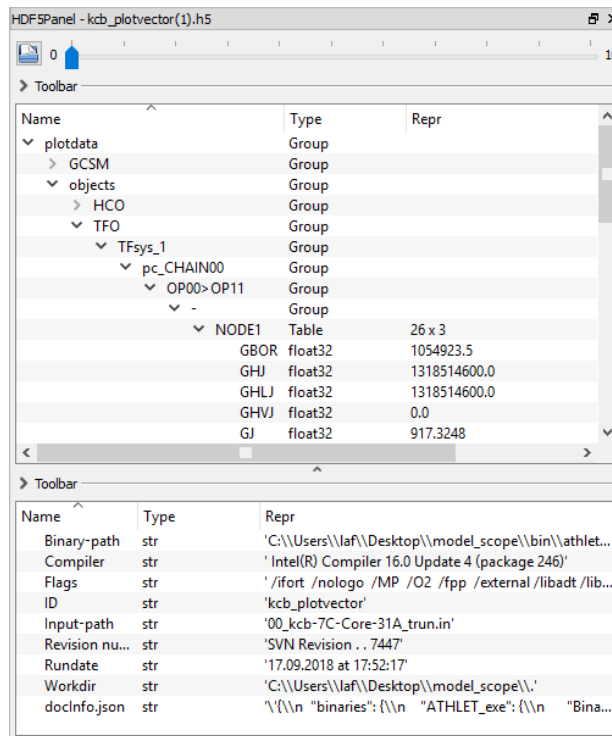


Abb. 4.62 Das HDF5Panel kann HDF5-Dateien einladen und stellt deren Inhalt hierarchisch dar – der Bereich unten zeigt die enthaltenen Metainformationen.

Aus den im Abschnitt 4.2.3.5 beschriebenen Gründen setzt ATLASneo bislang voraus, dass verwendbare Simulationsdaten im HDF5-Format vorliegen. Da sich HDF5 in den letzten Jahren allerdings zum De-facto-Standardformat großer Datenbestände entwickelt hat, welches mittlerweile auch von ATHLET direkt erzeugt werden kann, stellt diese Voraussetzung keine große Einschränkung dar, garantiert aber effiziente Datenverarbeitung innerhalb von ATLASneo. Um zukünftig auch Datenbestände anderer Formate verarbeiten zu können, müssen den Anwendern entsprechende Konverter bereitgestellt werden.

4.2.3.7 Das Steuermodul für Online-Simulationen

Die Möglichkeiten des Applikationsrahmens sowie des Drag-and-Drop-Protokolls konnten als erstes durch die Entwicklung des MonitorWidgets, einem Steuermodul für Online-gesteuerte ATHLET-Simulationen, praktisch genutzt und evaluiert werden (siehe Abb. 4.63). Dieses Funktionsmodul erlaubt es Online ATHLET-Simulationen zu starten, diese zeitschrittgenau zu steuern und auf den Datenbestand der Simulation zuzugreifen.

Es fungiert als Datenquelle und kann allen weiteren Funktionsmodulen ein sehr großes Spektrum an unterschiedlichen Simulationsdaten bereitstellen.

Um möglichst flexibel und umfangreich auf Simulationsdaten zuzugreifen und diese als Datenquelle für die Visualisierung nutzen zu können, erlaubt das Steuermodul das Einladen und Starten von ATHLET-basierten Simulationscodes. Hierzu können über die Oberfläche sowohl das zu verwendende Binary (die shared library von ATHLET), als auch der Datensatz und alle weiteren erforderlichen Startparameter angegeben werden. Während der Simulation ist dann der Zugriff auf die zugänglichen Daten der Simulation als auch auf die protokollierten Plotdaten möglich. Diese werden unter Verwendung des im Abschnitt 4.2.3.5 beschriebenen Moduls HDF5Writer als zusätzliche Ausgabedatei gespeichert.

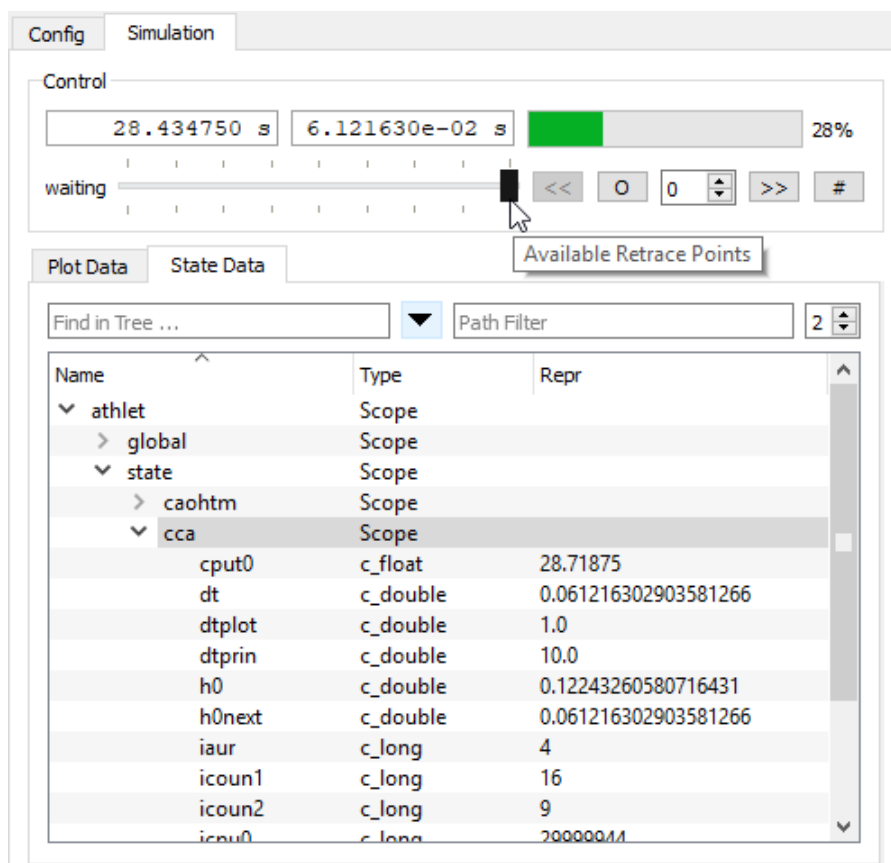


Abb. 4.63 Steuermodul für Online ATHLET-Simulationen: Zusätzlich zu Steuerelementen werden der aktuelle Simulationszustand (State Data) sowie Plotdaten (Plot Data) in Form eines Baumes dargestellt.

Nach dem Start und dem Einlesen des Datensatzes wird die Simulation angehalten und der im Simulator zugänglich gemachte Simulationszustand einem für ATHLET-Daten-

Scopes geeigneten `DataModel`-Objekt (`ScopeModel`) zugeordnet. Ein `DataModel` implementiert alle grundlegenden Aktionen des Datenzugriffs und fungiert somit als Adapter zwischen der Struktur der verknüpften Daten und einem Anzeigeelement der Bedienoberfläche. Im Fall des `ScopeModels` sind das die hierarchische Struktur der Simulordaten und der Baumdarstellung (`TreeView`) im Tab „State Data“ des `MonitorWidgets`. Dieses Anzeigeelement stellt die durch das `ScopeModel` bereitgestellten Daten hierarchisch dar und erlaubt diese interaktiv zu durchsuchen, zu filtern und ggf. auch zu verändern. Die dargestellten Daten werden dabei nicht zwischengespeichert, sondern bei jeder notwendigen Aktualisierung der Anzeige neu aus dem Speicherbereich des Simulators gelesen. Dadurch müssen keine weiteren Vorkehrungen getroffen werden, um die angezeigten Werte zu aktualisieren und vor allem die Anzeige mit dem aktuellen Zustand der Simulation konsistent zu halten. Was im ersten Moment aufwändig klingt, ist ein großer Vorteil für Speicherbedarf und Performance: Da die Ansicht (`TreeView`) nur die aktuell sichtbaren Daten vom `ScopeModel` anfordert, bewegen sich die notwendigen Zugriffe in engen Grenzen und entfallen vollständig sobald die `TreeView` ausgeblendet oder verdeckt wird.

Das gleiche Vorgehen wird für die Darstellung der aktuellen Plotdaten angewendet. Auch diese werden in einem eigenen Tab („Plot Data“) durch eine eigene `TreeView` hierarchisch dargestellt. Da den Plotdaten allerdings eine HDF-Datei als Datenstruktur zugrunde liegt (siehe Abschnitt 4.2.3.5) musste hierfür eine angepasste Variante des `DataModels` (`HDF5Model`) implementiert werden. Das `HDF5Model` realisiert die Datenzugriffe passend für HDF5-Dateien und kann alle gängigen HDF5-Elemente als hierarchische Struktur verfügbar machen. Hierdurch können auch HDF5-Dateien ohne weitere Änderungen in den Anzeigeelementen (`TreeViews`) von `ATLASneo` zur Verfügung gestellt werden.

Die in der Oberfläche vorbereiteten Eingabefelder für die Such- und Filterfunktionen erlauben es dem Benutzer bestimmte Daten schnell zu lokalisieren und auch sich auf Teilbereiche des Simulationszustandes zu konzentrieren. Da hierbei als Eingaben auch reguläre Ausdrücke akzeptiert werden, ist es ein wertvolles Werkzeug zur Suche und Auswahl von Simulationsdaten die dann in anderen Funktionsmodulen weiterverarbeitet und graphisch aufbereitet werden. Da diese Mechanismen im eingesetzten `TreeView` implementiert wurden und auf allen `DataModels` arbeiten, funktionieren diese Such- und Filterfunktionen ebenso in der Anzeige der Plotdaten oder in Darstellungen von HDF5-Daten aus anderen Quellen.

Der Ablauf der durch das `MonitorWidget` gestarteten Simulation kann im Folgenden frei kontrolliert werden. Diese Kontrolle wird dem Benutzer durch Buttons ermöglicht und erlaubt das Unterbrechen, Fortsetzen (nach Zeitschrittzahl oder unbegrenzt), das Setzen von und der Rücksprung zu Restartpunkten sowie den Abbruch der Simulation. Der Simulationsfortschritt und der aktuelle Simulationszustand werden währenddessen in der Oberfläche fortlaufend angezeigt.

Das `MonitorWidget` ist darauf ausgelegt, um Simulationen einfach starten zu können und deren Plot- und Zustandsdaten den anderen Funktionsmodulen bereitzustellen. In den `TreeViews` dargestellte Simulationsgrößen aus ATHLET können mit der Maus markiert und in andere Anwendungsbereiche gezogen werden. Hierzu implementiert das `MonitorWidget` das im Abschnitt 4.2.3.4 beschriebene Drag-and-Drop-Protokoll, indem es die entsprechenden Datenreferenzen der gewählten Variablen zur Verarbeitung bereitstellt. Zur Neuverarbeitung der Daten implementiert es das Signal `signal/dragDataChanged`, auf das sich andere Funktionsmodule verbinden können. Dieses Signal wird nach jeder Veränderung des Simulationszustandes (z. B. abgeschlossener Zeitschritt) ausgelöst und bewirkt so die Aktualisierung aller verknüpften Funktionsmodule.

4.2.3.8 Das Plotmodul

Das im vorherigen Abschnitt beschriebene Steuermodul erlaubt dem Anwender alle möglichen Werte einer Online-Simulation zu überwachen. Um deren zeitliche Entwicklung visuell verfolgen zu können wurde ein einfach zu konfigurierendes Plot-Modul (Abb. 4.64) implementiert. In Zusammenarbeit mit dem `MonitorWidget` können die zu plottenden Simulationsgrößen hierbei sehr einfach über den Drag-and-Drop-Mechanismus festgelegt werden. Auch die Abszissenachse kann dadurch gewählt werden, was den Anwendern eine sehr flexible Plotmöglichkeit bietet. Die Ansicht des Plotfensters ist interaktiv mit der Maus bedienbar und ermöglicht durch Zoom, Verschiebung und Skalierung die freie Anpassung des sichtbaren Plotbereichs. Die jeweils ausgewählten Simulationsgrößen werden in der Legende aufgelistet und fortlaufend aktualisiert geplottet. Das Plotmodul ist darauf ausgelegt alle HDF5-basierten Zeitreihen zu verarbeiten. Diese können auch durch das in Abschnitt 4.2.3.6 beschriebene `HDF5Panel` zur Verfügung gestellt werden. Dadurch kann das Plotmodul genauso für die Erstellung von Trends abgeschlossener Simulationen genutzt werden. Wie in Abb. 4.64 dargestellt,

können gleichzeitig mehrere Funktionsmodule als Datenquelle zum Einsatz kommen, um die Trends flexibel zusammenzustellen.

Zwar bietet das Plotmodul bereits eine große Bandbreite an Möglichkeiten sowohl Online-Simulationen durch live-Plots zu verfolgen als auch andere HDF5-Daten durch Trends auszuwerten. Dennoch fehlen noch wichtige Funktionen, um es produktiv einsetzen zu können. Besonders durch die ersten Nutzerrückmeldungen wurde klar, dass das Plotmodul zukünftig vor allem um einfach zugängliche Konfigurationsmöglichkeiten, wie z. B. das Festlegen von Farben, Schriftarten und Achsenbeschriftungen, erweitert werden muss.

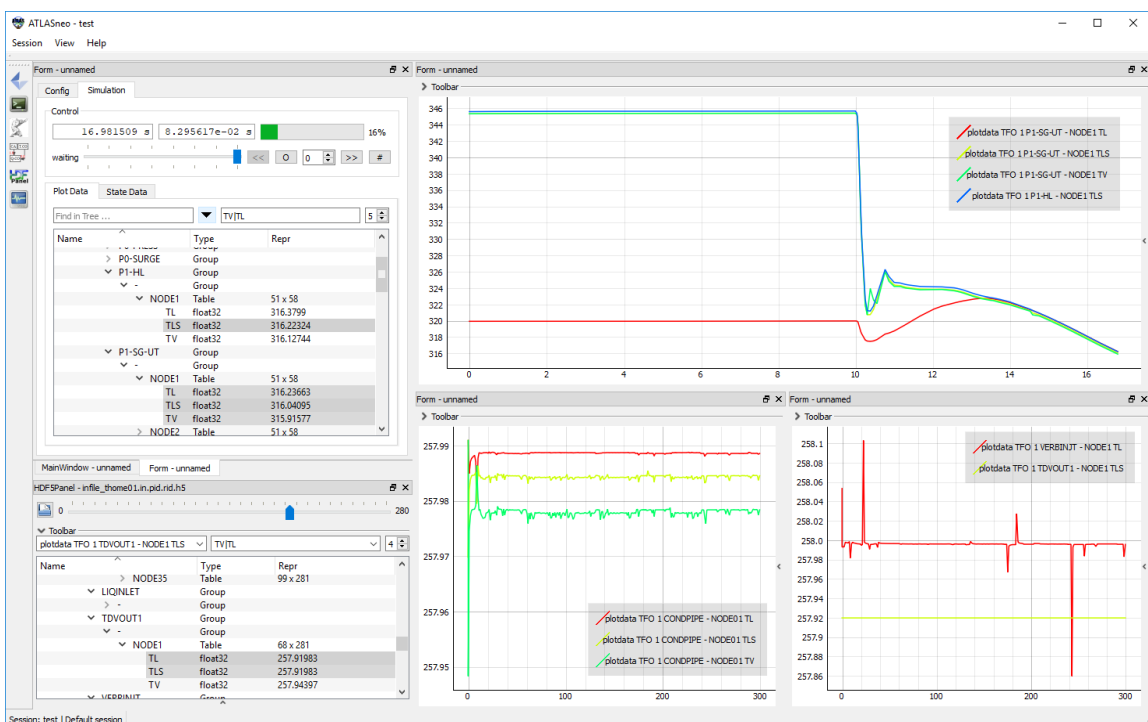


Abb. 4.64 Das Plotmodul im Einsatz: Die zu plottenden Größen wurden im Plotdatenbaum einer Online-Simulation (MonitorWidget) sowie einer HDF5-Datei (HDF5Panel) ausgewählt und per Drag-and-Drop den Plotfenstern zugeteilt.

4.2.3.9 Darstellung dynamisierter SVG-Bilder

Viele physikalische Zusammenhänge können dem Benutzer am besten durch eine bildliche Darstellung des Anlagenmodells verdeutlicht werden. Dabei werden Simulationswerte, wie Druck und Temperatur, in einer vorgegebenen geometrischen Szene abgebildet und z. B. als Farbe oder Position eines Geometrieobjektes visualisiert.

Konzeption

Die für die Abbildung physikalischer Werte benötigte Spezifikation der Geometrie der Bildelemente erfolgt in ATLASneo mittels einer SVG-Datei. Da es sich bei dem SVG-Format um einen etablierten und weithin unterstützten Standard handelt, kann sowohl für die Erstellung dieser Dateien als auch für die Darstellung der Bilder auf eine Fülle bestehender Werkzeuge zurückgegriffen werden. So ist es naheliegend für die Interpretation und Darstellung dieser Bilder auf der im Qt-Framework vorhandenen Funktionalität aufzusetzen. Der Benutzer profitiert von der Verwendung eines Standardformates, indem es ihm letztlich freigestellt ist, in welcher Anwendung er die seiner Visualisierung zu Grunde liegende Szene modelliert, solange diese als SVG /SCA 11/ exportiert werden kann.

Implementierung

Natürlich muss ATLASneo noch vielen weiteren Anforderungen als der schlichten Darstellung dieser Bilder gerecht werden. Darum wurde ein GUI-Element (Widget) entwickelt, welches bereits viele dieser Anforderungen zentral implementiert und das in Funktionsmodulen verwendet werden kann, um interaktive oder dynamisierte Inhalte darzustellen. Sehr viele der hier benötigten Funktionen zu Darstellung, Interaktion und Dynamisierung finden seit geraumer Zeit auch auf vielen Webseiten Verwendung, wodurch sich eine Vielzahl an hilfreichen Techniken und unterstützenden Bibliotheken entwickelt hat. Aus diesem Grund zielt das implementierte Widget auf eine möglichst einfache Nutzung webbasierter Techniken ab und stellt den Funktionsmodulen bereits einiges an Unterstützung zur Verfügung:

- Bereits viele der individuellen Bedürfnisse eines Funktionsmoduls können über einen darauf angepassten HTML-Rahmen realisiert werden. Wird dieser als Teil des Modul-Packages bereitgestellt, wird der Rahmen automatisch erkannt und geladen.
- Für den Datenaustausch zwischen dem Funktionsmodul und im HTML-Rahmen enthaltenen JavaScript /WIK 17a/-Code stehen Qt-Signale und Transferobjekte zur Verfügung, welche den direkten Austausch von Daten und eine ereignisgesteuerte Aktualisierung der dargestellten Bilder ermöglichen.
- Das Verbinden von im Funktionsmodul definierten Signalen mit den in JavaScript implementierten Slot-Funktionen erfolgt automatisch. Der im Widget integrierte Lademechanismus analysiert hierzu die eingeladenen JavaScript-Routinen. Für

Routinen (benannt nach dem Namensschema `slot_<name>`) wird versucht, diese mit einem gleichnamigen Signal im Funktionsmodul zu verbinden. Die Reaktion auf Ereignisse kann dadurch sehr einfach und flexibel implementiert werden.

- Zur Anpassung der Ansicht wurden die mittlerweile gängigen Mausinteraktionen `WheelZoom` (Zoomen über das Mausrad) und `DragPan` (Verschieben der Ansicht durch Ziehen) implementiert. Dadurch kann der Benutzer den sichtbaren Bildausschnitt schnell und flexibel anpassen.

4.2.3.10 Bilddarstellung von Simulationen mit EIDOS

Das ATLASneo-Modul "EIDOS" (Enhance Images with Dynamic Object Status, vgl. griechisch εἶδος: das zu Sehende) dient primär zur Visualisierung der zeitabhängigen Rechenergebnisse von GRS-Rechenprogrammen aus dem AC²-Paket anhand von zweidimensionalen grafischen Modellen, soll aber auch als grafische Benutzerschnittstelle zur Steuerung von interaktiven Simulationen verwendbar sein. Als Simulationssoftware wird zurzeit nur ATHLET und sein traditionelles Ausgabeformat (Pd-, Key- und Gr-Dateien) unterstützt. EIDOS wird in Python auf Basis der Fensterbibliothek Qt/PySide /PYS 15/ programmiert. Zur Darstellung und zum Skripten der SVG-Bilder kommt die Klasse `QWebView` zum Einsatz, die als Schnittstelle zur Browser-Implementierung "Webkit" (<https://webkit.org/>) dient. Webkit beinhaltet eine JavaScript-Umgebung, die ECMAScript-Code der Version 5 interpretiert, was eine Veränderung von Webseiten und den enthaltenen SVG-Elementen zur Laufzeit erlaubt. EIDOS wurde so angelegt, dass es sowohl alleinstehend als auch im Rahmen von ATLASneo lauffähig ist.

EIDOS verwendet Bilddateien im XML basierten SVG-Format (<https://www.w3.org/Graphics/SVG/>), wie sie z. B. über den Umweg der Konvertierung durch `apg2svg` mit der ATHLET Input Grafik oder auch manuell mit generischen Editoren wie dem quelloffenen Inkscape (<http://www.inkscape.org>) erstellt werden können. Die Schnittstelle zwischen ATHLET-Modell und EIDOS-Bildern basiert auf Namen und Nummerierungen. SVG-Dateien müssen bestimmten Formatvorgaben entsprechen: die grafischen Bestandteile von Thermo-Fluid- und Wärmeleitungsobjekten (TFOs und HCOs) sind in benannten und mit bestimmten Attributen versehenen SVG-Gruppen zusammengefasst. Die Obergruppe eines ATHLET-Objekts zeichnet sich durch die HTML-Attribute "class" und "id" aus, wobei ersteres den Wert "object" annimmt und die ID dem Objektnamen im ATHLET-Modell entspricht. Die hierarchisch darunter angesiedelten Gruppen der Kontrollvolumina haben die Klasse "node" und das EIDOS spezifische Attribut "nodeld", das

0-basiert die Rechenknoten des Objekts aufzählt. Darin enthaltene Flächenelemente, die zur Farbdarstellung von Modellvariablen dienen sollen, müssen als "fill"-Attribut den Wert "inherit" haben, damit dieser nachher durch Zuweisung einer Füllfarbe an die Kontrollvolumengruppe überschrieben werden kann.

Der Benutzer ruft über Taster Dialoge zum separaten Laden von Bildern und Simulationsdaten auf. Beim Lesen der SVG-Datei identifiziert das Modul die mit Klassenattributen als Modellobjekte deklarierten Grafikelemente. Während des darauffolgenden Ladens der Simulationsdaten findet ein Abgleich dieser Objekte mit dem Rechenmodell statt. Bei Unstimmigkeiten bezüglich Namen oder Kontrollvolumenanzahl deaktiviert EIDOS die entsprechenden Bildbestandteile und gibt eine Warnung aus. Des Weiteren stellt EIDOS soweit möglich anhand der Geometrie und der Informationen in der ATHLET-Gr-Datei automatisch Verbindungen zwischen Einbauten wie Pumpen und Ventilen mit ihren Daten im ATHLET-Plotvektor her. In der alten ATLAS-Implementierung waren diese grafischen Elemente in Ausgabedateien von AIG dagegen bisher inaktiv, sofern sie nicht nachträglich mit dem APG-Editor jeweils fest mit einer Variablen verknüpft wurden.

Nach der Dateneingabe stehen dem Benutzer Auswahlfelder zur Selektierung der farbkodiert darzustellenden TFO und HCO-Variablen zur Verfügung. Sie beinhalten nicht wie das alte ATLAS eine vorher in den Bildern festgelegte Liste an Einträgen, sondern stellen alle in der Simulation für den entsprechenden Objekttyp definierten Größen bereit, so dass die SVG-Datei bei Modellerweiterungen nicht editiert werden muss. Bei der Aktivierung einer Variablen wird ein zugehöriger Farbbalken eingeblendet. Dieser ist mit dem Mauszeiger verschieb- und vergrößerbar und kann durch ein per rechter Maustaste aufgerufenes Kontextmenü weiter konfiguriert werden (Abb. 4.65). So lässt sich z. B. die dargestellte Farbskala wählen. Voreingestellte, häufig verwendete Paletten, die dem Programm "ParaView" und der Python-Bibliothek "Matplotlib" entlehnt wurden, sind in der Datei "colourMaps.yml" im ASCII-Format YAML hinterlegt (siehe <https://yaml.org/>). Zusätzliche die Farbbalken betreffende Einstellungen sind das Zahlenformat der zugehörigen Werteskala und der Modus, mit dem deren Intervallgrenzen bestimmt und aktualisiert werden: benutzerdefiniert, auf den derzeitigen oder den jeweils aktuellen Zeitschritt bezogen oder auf zeitlich globale Minimal- und Maximalwerte der dargestellten Variablen normiert.

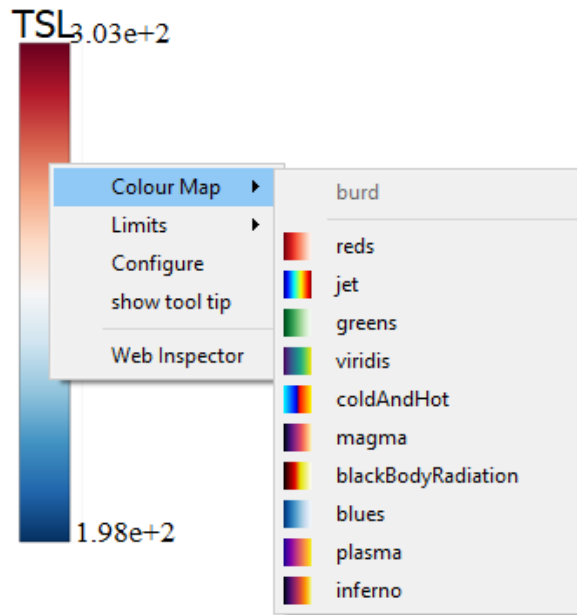


Abb. 4.65 Farbbalken mit Kontextmenü

Ein verknüpftes Bedienelement-Paar aus Schieberegler und Ganzzahl-Eingabefeld dient im Einzelmodus zur interaktiven Auswahl des Simulationszeitschritts. Neben den Farbbalken reagieren auch der Bildhintergrund, Einbauten und TFO/HCO-Objekte auf Mausinteraktionen. Beim Überstreichen eines Kontrollvolumens hebt EIDOS dessen Umrandung farblich hervor. Wurde per Kontextmenü die Option "show tool tip" aktiviert, erscheint zusätzlich der aktuell dargestellte Variablenwert oder die Kontrollvolumen-ID. Das Kontextmenü identifiziert außerdem beim Auslösen über einem Simulationselement in seiner Kopfzeile den Modellnamen des entsprechenden Objekts und gegebenenfalls den Modellindex des Kontrollvolumens.

Neben der Einfärbung der abgebildeten Simulationsobjekte anhand der Daten des aktuellen Zeitschritts und der zugehörigen Farbskala existieren weitere automatische Dynamisierungen: erkennt EIDOS, dass ein Objekt mit einem ATHLET Mischspiegelmodell ausgestattet wurde, stellt das Modul dieses als Füllstand dar. Das dazu über die entsprechenden Kontrollvolumina gelegte Grafikelement wird ebenfalls gemäß dem aktuell ausgewählten Variablennamen eingefärbt, nur wird dafür der entsprechende Wert aus dem ATHLET-"MIXLEVEL"-Modell verwendet. Technisch wird dieser Effekt durch ein anhand der ATHLET-Variablen ZML in seiner Höhe angepasstes SVG-Rechteck implementiert, das als "clip-path"-Attribut eine Referenz auf die Umriss des Simulationsobjekts verwendet. Durch Benutzerinteraktion können außerdem zeitlich animierte Textdarstellungen von Simulationenwerten erzeugt werden: der Kontextmenü-Eintrag

"add dynamic text" öffnet einen Auswahldialog für Variablen, dessen Wertebereich gegebenenfalls auf die unter dem Mauszeiger befindliche Komponente (TFO, HCO, Pumpe oder Ventil) eingeschränkt ist (Abb. 4.66). Außerdem stellt EIDOS gegebenenfalls den Kontrollvolumenselektor auf den richtigen Wert ein. Die Textobjekte sind verschieb-, editier- und entfernbar und können mit zusätzlichen statischen Annotationen versehen werden. Wählt der Benutzer die Option "add node-bound dynamic text" passt sich die Darstellung an die aktuelle ausgewählte Variable der entsprechenden Objektklasse (TFO oder HCO) an.

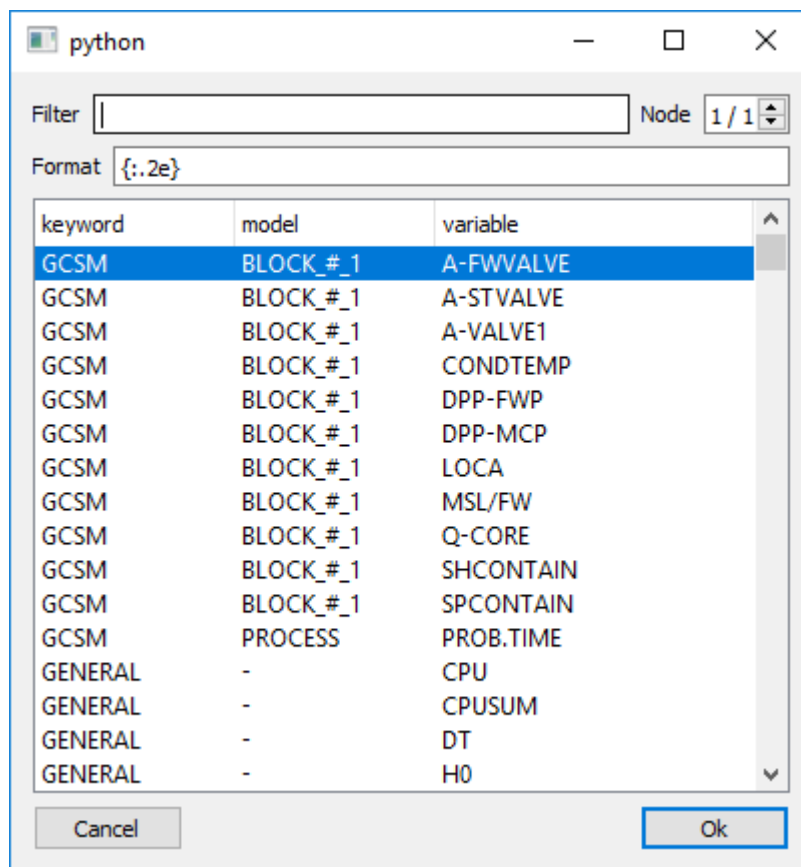


Abb. 4.66 Variablenauswahl für dynamisches Textobjekt

Durch das Rendern mit einem Browser ist die Darstellung der SVG-Elemente dank der semantischen Auszeichnung mit Klassen und IDs feingranular über "cascading stylesheets" (CSS, <https://www.w3.org/Style/CSS/>) anpassbar. Diese Konfigurationen sind in der Textdatei "eidos.css" abgelegt und können editiert und per Tastaturkommando (Strg-c) dynamisch nachgeladen werden. Programmlogische Einstellungen befinden sich in der Datei "eidos.yml", wo der Benutzer unter anderem bevorzugte Farbskalen für bestimmte Variablennamen hinterlegen kann.

In Abb. 4.67 ist beispielhaft die Anwendung von EIDOS auf den in der ATHLET-Distribution mitgelieferten „sample1“-Datensatz dargestellt. Die verwendete SVG-Datei wurde durch Konvertierung des APG-Bildes aus der ATHLET Input Grafik mit apg2svg gewonnen. Zu erkennen sind dynamisch mit den Variablen TSL und AV eingefärbte HCO und TFO-Objekte mit den entsprechenden Farbbalken. Das Kontrollvolumen unter dem Mauszeiger (nicht sichtbar) ist grün umrandet. Weil die Option „show tool tip“ aktiviert ist, wird auch eine Textdarstellung des Variablenwerts im Rechenknoten dargestellt. Mehrere dynamische Textobjekte wurden interaktiv an den Pumpen in den Objekten S1-AX-FW und P1-CL hinzugefügt, die unterschiedliche Variablen anzeigen. Im TFO P0-PRESS erkennt man die Füllstandsdarstellung eines Gemischspiegel-Modells. Daneben sind Textobjekte, die untereinander die Variable AV und den analogen Wert des „mixture level“-Modells anzeigen.

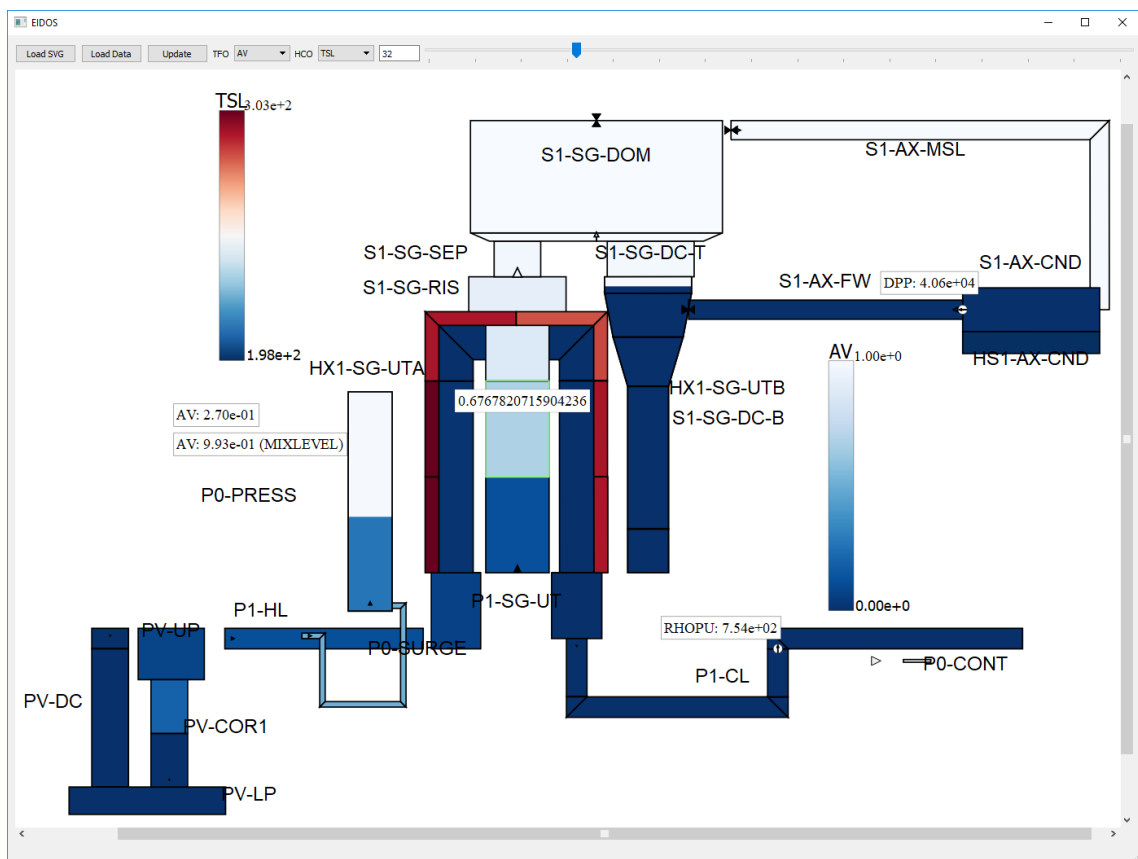


Abb. 4.67 ATHLET "sample1" in EIDOS

Technisch wurde EIDOS daraufhin angelegt, dass die SVG/JavaScript-Seite so gekapselt ist, dass sie keinerlei Informationen über das konkrete Simulationsmodell hat und sich nur auf die Implementierung der Darstellung beschränkt, deren Details sie ihrerseits bis auf Schnittstelleninformationen nicht nach außen dringen lässt. Zurzeit sind Teile von

EIDOS rein auf die Bedürfnisse von ATHLET zugeschnitten. Sie sind in Zukunft so zu abstrahieren und zu erweitern, dass auch die Anforderungen der anderen AC²-Komponenten abgedeckt werden können. Eine weitere Einschränkung des Moduls in seiner aktuellen Version ist, dass der Applikationsrahmen nicht als Datenquelle verwendet werden kann, sondern nur bereits berechnete ATHLET-Läufe im traditionellen Datenformat. Das liegt unter anderem daran, dass ATHLET in sein neues, HDF5 basierte Ausgabenformat, das auch ATLASneo zugrunde liegt, zurzeit noch nicht alle für EIDOS relevanten Größen exportiert.

Bisher sind neben den bisher erwähnten Beschränkungen folgende bildbezogene Fähigkeiten des alten ATLAS-Programms noch nicht implementiert:

- ATHLET-CD spezifische Spezialbilder (z. B. AIDA, MEWA, LHEAD, SHAPIRO).
- benutzerdefiniertes Scripting und UI-Elemente (bisher durch APG-Dynamikbefehle): hierfür sollen eine JavaScript-Bibliothek entwickelt und entsprechende Eingriffspunkte ("hooks") in den Programmablauf geschaffen werden.
- Speichern der Benutzerkonfiguration von dynamischen Bildelementen, z. B. Farbbalken und Textobjekte: angedacht ist das Schreiben einer Zip-Datei, die das Original-SVG und in einer JSON-Datei die Einstellungen speichert.
- Pegel-Darstellung bzgl. ATHLET "collapse level".
- Auslösen von Aktionen in anderen ATLASneo-Modulen, wie z. B. die Anzeige von Zeitverlaufdiagrammen eines Kontrollvolumens, das Öffnen des Eintrags eines Objekts im Variablenbaum oder die Simulationssteuerung.
- Zoom auf einen definierten Bereich: im Moment ignoriert die Vergrößerungsfunktionalität noch die aktuelle Mauszeigerposition. Der interessierende Bereich muss nach einer Zoom-Änderung zumeist durch Scrollen wieder in das Sichtfenster gebracht werden.
- Umbenennung von Simulationsvariablen.

4.2.3.11 GCSM-Netzwerke

Eine wichtige Anforderung an ATLASneo ist die Darstellung und Analyse der Leittechnik. Dieser große und sehr dynamische Teil eines Anlagenmodells erfordert durch seine weitverzweigten Signalnetzwerke besondere Formen der Darstellung. Durch die hohe Flexibilität und Komplexität in der Leittechnikmodellierung macht es nur in Ausnahmefällen

Sinn, die zur Visualisierung benötigte Geometrie vorab zu modellieren. Deshalb wurde hierfür ein spezielles Funktionsmodul GCSMViz implementiert, welches die in einer Simulation vorhandene Leittechnik automatisch analysiert und deren Signalnetzwerk als gerichteten Graph visualisiert. Wie in Abb. 4.68 ersichtlich, werden die Signalwerte dabei als dynamisierter Text an den Blockeingängen dargestellt und bei jeder Veränderung aktualisiert.

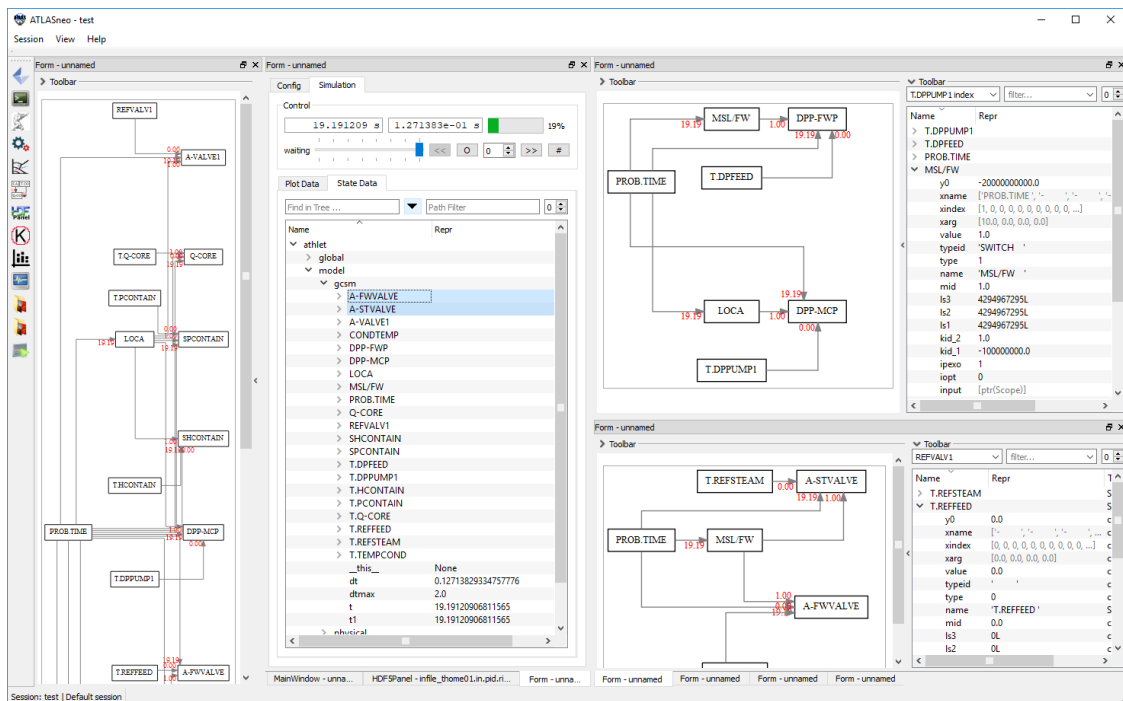


Abb. 4.68 Online-Simulation in ATLASneo: Verwendung des Funktionsmoduls GCSMViz und des Plotmoduls zur Analyse der Leittechnik sowie des zeitlichen Verlaufs von Zustandsvariablen

Natürlich baut auch dieses Funktionsmodul auf den Schnittstellen des Applikationsrahmens und des Drag-and-Drop-Protokolls auf. Seine Entwicklung diente insbesondere auch dazu, die Konzepte für Bedienung, SVG-Darstellung und Dynamisierung per JavaScript [WIK 17a/ zu evaluieren. GCSMViz profitiert von den Möglichkeiten der zentral implementierten Konzepte des Anwendungsrahmens und konnte entsprechend kompakt realisiert werden. Durch die Verwendung von Drag-and-Drop konnte die Bedienung rein mausbasiert gestaltet werden, wodurch bisher keine zusätzlichen Kontrollelemente in der Oberfläche des Funktionsmoduls notwendig sind.

Um ein Signalnetz darzustellen, kann eine freie Auswahl von GCSM-Signalen im Steuermodule gewählt und auf das Modul gezogen werden. Beim anschließenden Aufbau des Graphen werden die Eingänge der gewählten Signale rekursiv zurückverfolgt und auch

diese in das Netz mit aufgenommen. Um die Entstehung eines bestimmten Signals zu analysieren, reicht es also aus alleine dieses zu wählen und GCSMViz wird es als Graph komplett mit allen Abhängigkeiten in einer zoombaren Ansicht darstellen.

Zur Unterstützung der Anwender bei der Signalanalyse bieten sich hier viele weitere Möglichkeiten an. Die zukünftige Implementierung solcher Funktionen sollte aber mit den Bedürfnissen der Benutzer abgestimmt werden und kann nach und nach erfolgen.

4.3 Reengineering der ATHLET Input Graphic AIG

4.3.1 Trennung der Programmteile

FORTRAN-DLL

Im Rahmen der Neuentwicklung der ATHLET Input Graphic AIG wurden Arbeiten zur Trennung der Programmteile, d. h. der alten vorliegenden Bedienoberflächenelemente (GUI) und des FORTRAN basierenden Programmkerns des AIG-Programmes, durchgeführt.

Der FORTRAN basierende Programmkern ist von ungenutzten Quellcode-Bestandteilen bereinigt worden und wurde für die weitere Verwendung im Rahmen des Reengineering-Ansatzes in eine dynamische Bibliothek (DLL) umgewandelt. Hiermit wurde die Grundlage für die Umstellung auf die plattformunabhängige Qt-Grafikbibliothek gelegt.

Schnittstellenanpassung zwischen FORTRAN-DLL und C++-Oberfläche

Der ursprüngliche Programmablauf (siehe Abb. 4.69) sah vor, dass die erstellte SVG-Datei durch die FORTRAN-DLL auf der Festplatte gespeichert und im Anschluss daran durch das Oberflächenprogramm gelesen werden soll.

Aus dieser Vorgehensweise würden jedoch intensive Speicher- und Lesezugriffe folgen, die die Performance des gesamten Programmablaufs nachteilig beeinflussen, da bereits geringfügige nutzerseitige Änderungen der Darstellung, beispielsweise durch Ein- und Ausblenden von TFO, zu Neuberechnung und damit verbunden zur Aktualisierung der SVG-Datei auf der Festplatte führen, die im Nachgang durch das Oberflächenprogramm erneut geladen werden müssen.

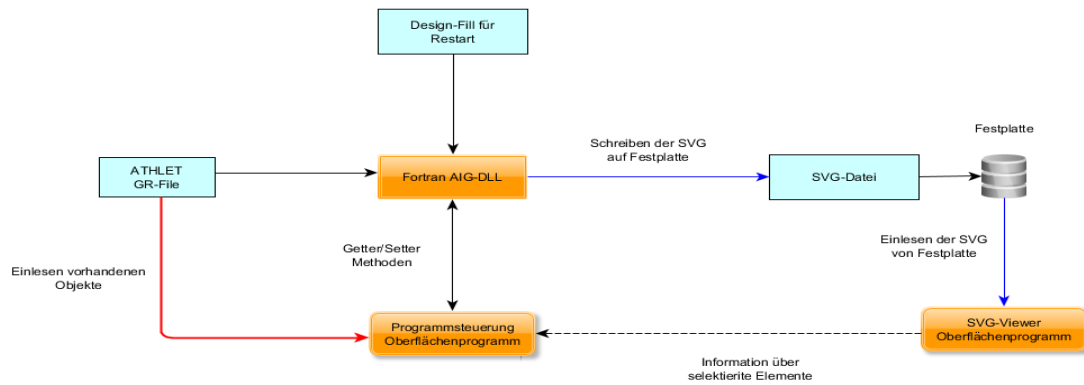


Abb. 4.69 Ursprüngliches Konzept für die neue AIG-Oberfläche

Zur Vermeidung der o. g. intensiven Festplatten-Schreibe- und -Lesezugriffe wurde das der DLL zugrunde liegende FORTRAN-Programm um ein Modul erweitert, welches eine Schnittstelle für den programminternen Austausch der SVG-Datei, d. h. innerhalb des Arbeitsspeichers, bereitstellt. Das überarbeitete Konzept für die neue AIG-Oberfläche ist in Abb. 4.70 dargestellt.

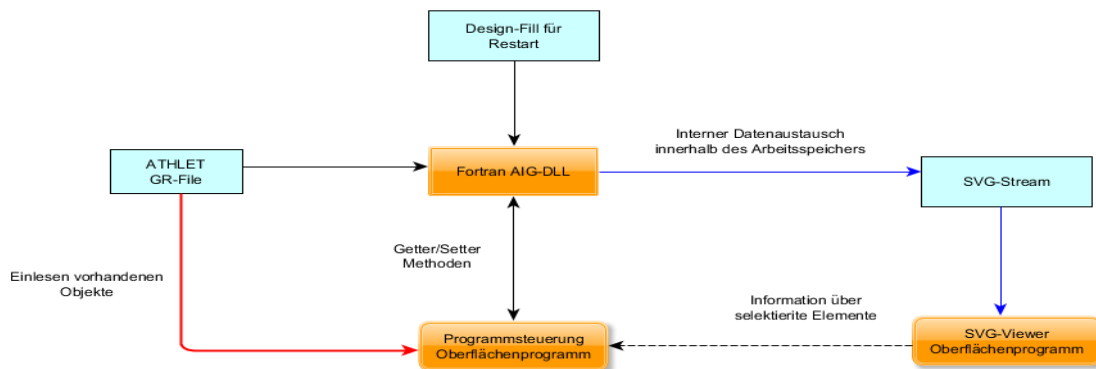


Abb. 4.70 Überarbeitetes Konzept für die neue AIG-Oberfläche

4.3.2 Implementierung einer GR-File und SVG-Parser-Klasse

Für die Ermittlung der für die Baumstruktur (siehe Abschnitt 4.3.3) und die grafische Darstellung notwendigen Informationen wurde eine entsprechende Parser-Klasse implementiert, die diese Daten aus dem durch das ATHLET-Programm erzeugte GR-File einmalig einliest.

Die geometrischen Informationen zu den verschiedenen grafischen Objekten werden durch die FORTRAN-DLL erzeugt und in einer entsprechenden SVG-Datei abgelegt. Da diese Informationen, bedingt durch die Nutzer-Modifikationen und -Interaktionen (z. B.

Verschieben einzelner Objekte, unterschiedliche Darstellungsformen usw.) einer kontinuierlichen Veränderung unterworfen sind, wurde neben der in der Qt-Bibliothek vorhandenen Shared SVG-Renderer-Klasse zusätzlich eine **SVG-Parser-Klasse** implementiert.

Die Notwendigkeit für die Erstellung dieser zusätzlichen SVG-Parser-Klasse ergab sich aus dem Sachverhalt, dass die verwendete Qt-Klasse (SharedRenderer) die benötigten Informationen bzgl. der individuellen Objektgeometrie nicht ermitteln kann und damit die objektspezifischen Begrenzungsrahmen (Bounding Box) nicht standardmäßig aus der erzeugten SVG-Datei zu bestimmen sind, mit der Folge, dass eine eindeutige Objektzuweisung und Selektion durch den Nutzer nicht möglich ist. Abb. 4.71 zeigt ein Beispiel für die Begrenzungsrahmen, wie sie durch die Qt-Klasse (SharedRenderer) bestimmt werden. Deutlich wird, dass die SVG-Rendererklasse der Qt-Bibliothek um jedes Objekt ein maximal einfassendes Rechteck als Begrenzungsrahmen zeichnet und somit der individuellen Geometrie bzw. der Linienführung des jeweiligen Zeichenobjektes nicht folgt.⁴ Somit besteht die Möglichkeit, dass die rechteckigen Begrenzungsrahmen größerer Zeichenobjekte die Begrenzungsrahmen der kleineren Zeichenobjekte überdecken, so dass diese durch den Nutzer über die Zeichenoberfläche nicht mehr selektierbar sind. Abb. 4.71 zeigt ein Beispiel für die problematischen Begrenzungsrahmen der Qt-Klasse (SharedRenderer). Der maximale rechteckige Begrenzungsrahmen (rote gestrichelte Linie) des Objektes 1 überdeckt den Begrenzungsrahmen von Objekt 2, wodurch Objekt 2 über die grafische Oberfläche nicht mehr durch den Nutzer selektierbar ist.

⁴ Bemerkung: Die verwendete SVG-Renderer-Klasse ist seit Qt-Version 4.0 Bestandteil der Qt-Bibliothek. Die aktuell verwendete Klasse unterstützt die statischen Eigenschaften von SVG 1.2 Tiny. Der Funktionsumfang beinhaltet jedoch nicht eine Unterstützung für DOM-Manipulationen bzw. EMCA (JavaScript, JScript oder ActionScript). Es handelt sich somit nicht um einen vollwertigen SVG-Renderer, wie er von Webbrowser- oder speziellen Graphikprogrammen angeboten wird. Die Implementierung alternativer Open Source SVG-Renderer außerhalb der Qt-Klassenhierarchie erschien kaum realisierbar.

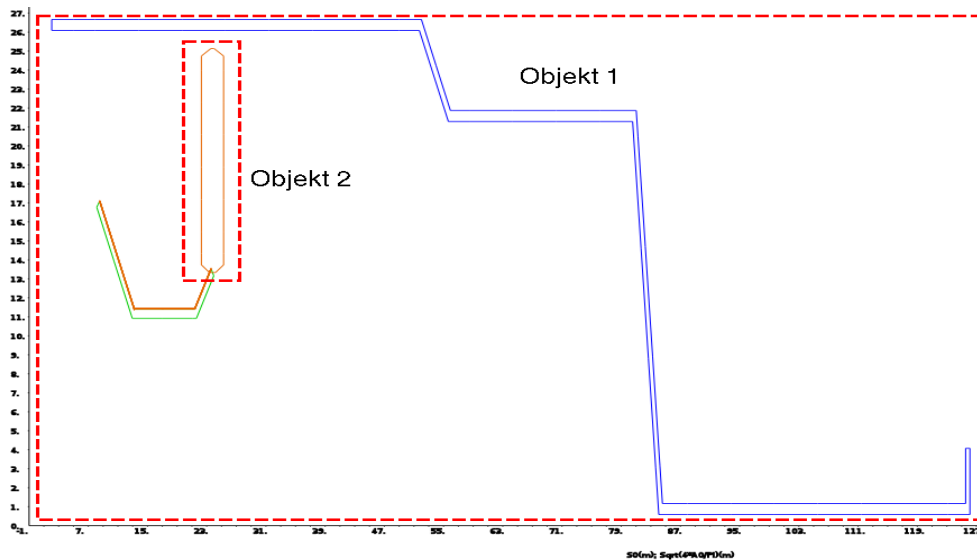


Abb. 4.71 Begrenzungsrahmen ermittelt durch die Qt-Klasse SharedRenderer

Implementierung eines angepassten Bezugsrahmens für die Objekte

Für die nutzerfreundliche Selektierbarkeit der einzelnen grafischen Elemente auf der Zeichenoberfläche wurde somit innerhalb der SVG-Parser-Klasse eine Methode entwickelt, die einen geometrieangepassten Begrenzungsrahmen (Bounding Box) um die einzelnen Zeichenobjekte erzeugt. Die hierfür notwendigen geometrischen Informationen sind in der SVG-Datei hinterlegt, welches ein erneutes, Zeichenpfad-orientiertes Einlesen (Par-sen) der einzelnen Objekte notwendig macht. Die Abb. 4.72 zeigt das Ergebnis der im-plementierten Methode. Durch den angepassten Begrenzungsrahmen (rote gestrichelte Linie) des Objektes 1 entsteht keine vollständige Überdeckung von Objekt 2.

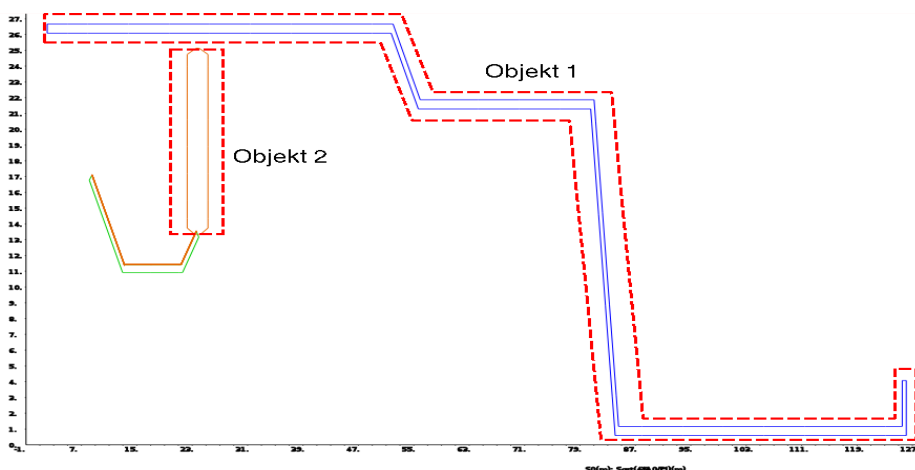


Abb. 4.72 Verbesserte Methode zur Bestimmung des Begrenzungsrahmens

Das überarbeitete Gesamtkonzept, das aus den oben beschriebenen Maßnahmen resultiert, ist in Abb. 4.73 dargestellt, d. h. das zusätzliche Einlesen der SVG-Datei /SCA 11/ für die Bestimmung des Begrenzungsrahmens.

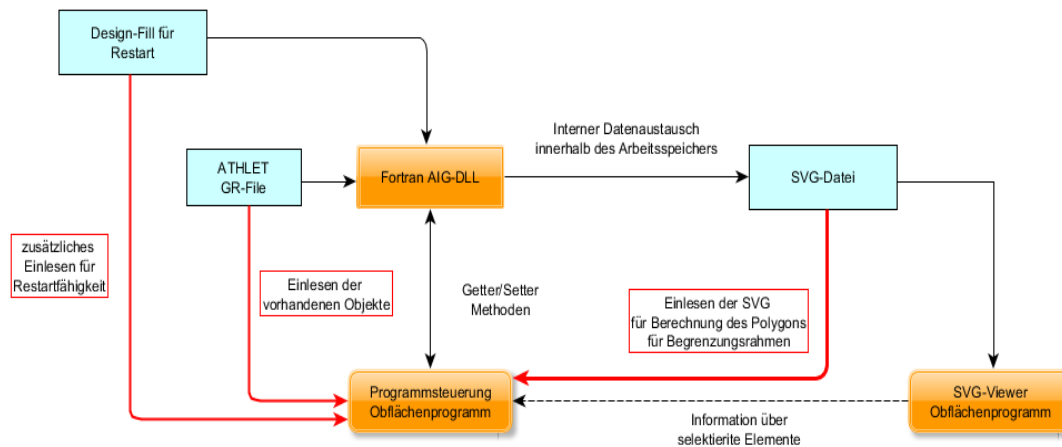


Abb. 4.73 Verbessertes Konzept der FORTRAN-DLL-AIG-Einbindung

4.3.3 Baumdarstellung der Objekte

Die Erstellung der neuen grafischen Oberfläche dient zur Ansteuerung des auf FORTRAN basierenden Programmkerns (.dll) und zur Visualisierung der von diesem Programmteil bereitgestellten SVG-Graphiken. Bestandteil der Arbeiten war die Neugestaltung und Überführung der ursprünglichen listenartigen Objektdarstellung in eine Baumdarstellung, um einen nutzerfreundlichen Zugriff auf die unterschiedlichen darzustellenden grafischen Objekte zu ermöglichen. Ziel hierbei war es, die unterschiedlichen grafischen Objekte, die in der exportierten SVG-Graphik enthalten sind, anhand einer dargestellten Objekt-Baumstruktur selektiv zu visualisieren und über eine Suchfunktion auffindbar zu machen.

Weiterhin sollten zusätzliche Informationen in dieser Baustruktur hinterlegt werden, die dem Nutzer ein schnelleres Auffinden von im ATHLET Output-File als fehlerhaft gekennzeichneten TFO-HCO und SOPHAEROS-Objekten ermöglicht.

Ferner sollten Angaben zu den jeweils verknüpften TFO-Objekten im Baum der HCO- und SOPHAEROS-Objekte dem Nutzer eine bessere Übersicht und eine schnellere Erstellung von grafischen Bildern ermöglichen. Neben Objektnummer K, die bereits in der alten AIG-Version aufgeführt wurde (Abb. 4.74), sind in der überarbeiteten AIG-Version (Abb. 4.75) folgende Zusatzinformationen hinterlegt worden:

TFO-Objekt:

- Anfangs- und Endnummer (ID) der zugehörigen Control Volumina (IILO, IIRO)
- Anfangs- und Endnummer (ID) der zugehörigen Junction (IJLO, IJRO)

HCO:

- die rechts- und linksseitig verknüpften TFO-Objekte (AOLH, AORH)

SOPHAEROS:

- die verknüpften TFO-Objekte (ATHNAM)

Wie Abb. 4.74 zeigt, werden neben den Objektnamen zusätzlich auch die zugehörigen Objektnummern aufgeführt. Für die Bearbeitung von Fehlern im Eingabedatensatz ist die Objektnummer jedoch ungeeignet, da die Fehlerausgabe im ATHLET-Output File sich jeweils auf die Control-Volumina und Junction-Nummer der einzelnen TFO-, HCO- und SOPHAEROS-Objekte bezieht.

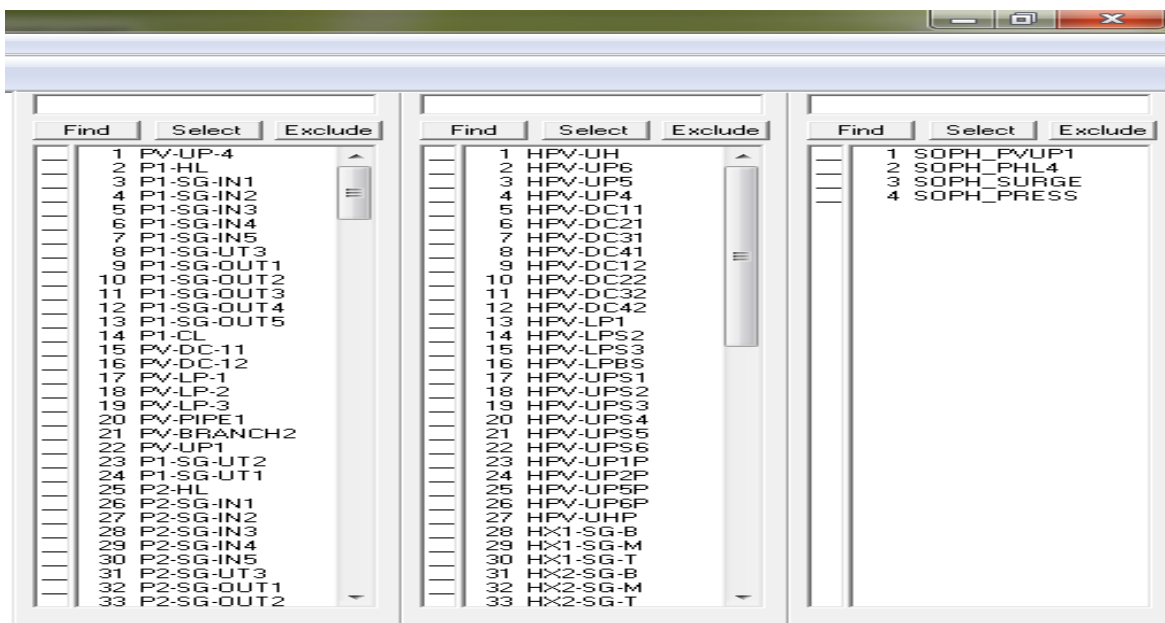


Abb. 4.74 Darstellung der TFO, HCO und SOPHAEROS-Objekte der alten AIG-Version

Durch die Darstellung und Sortiermöglichkeit der Control Volumina bzw. Junction Nummern in der aktualisierten AIG-Version (Abb. 4.75), kann nun die im Output-File als fehlerhaft gekennzeichneten Control Volumina bzw. Junction Nummern schneller aufgefunden und dem zugehörigen Objekt zugeordnet werden. Die somit bereitgestellte Verknüpfung zwischen Control Volumina bzw. Junction Nummer und dem Objekt-

Namen ermöglicht dem Nutzer eine schnellere Korrektur des betroffenen Objektes im ATHLET-Eingabedatensatz.

Überarbeitung der Suchfunktion

Wie in der bisherigen AIG-Version können die einzelnen Objekte über eine Suchfeldeingabe aufgefunden werden. In der überarbeiteten AIG-Version wurden die drei individuellen Suchfeldmasken für TFO, HCO und SOPHAEROS-Elemente durch eine Suchfeldmaske ersetzt (Abb. 4.75). Die individuelle Wahl des zu durchsuchenden Baumes erfolgt nun über die Selektion der Check-Boxen. Des Weiteren ist die Anzeige der aufgefundenen Objekte durch eine Filterfunktion überarbeitet worden, so dass ausschließlich diejenigen Objekte angezeigt werden, die die eingegebene Suchphrase enthalten.

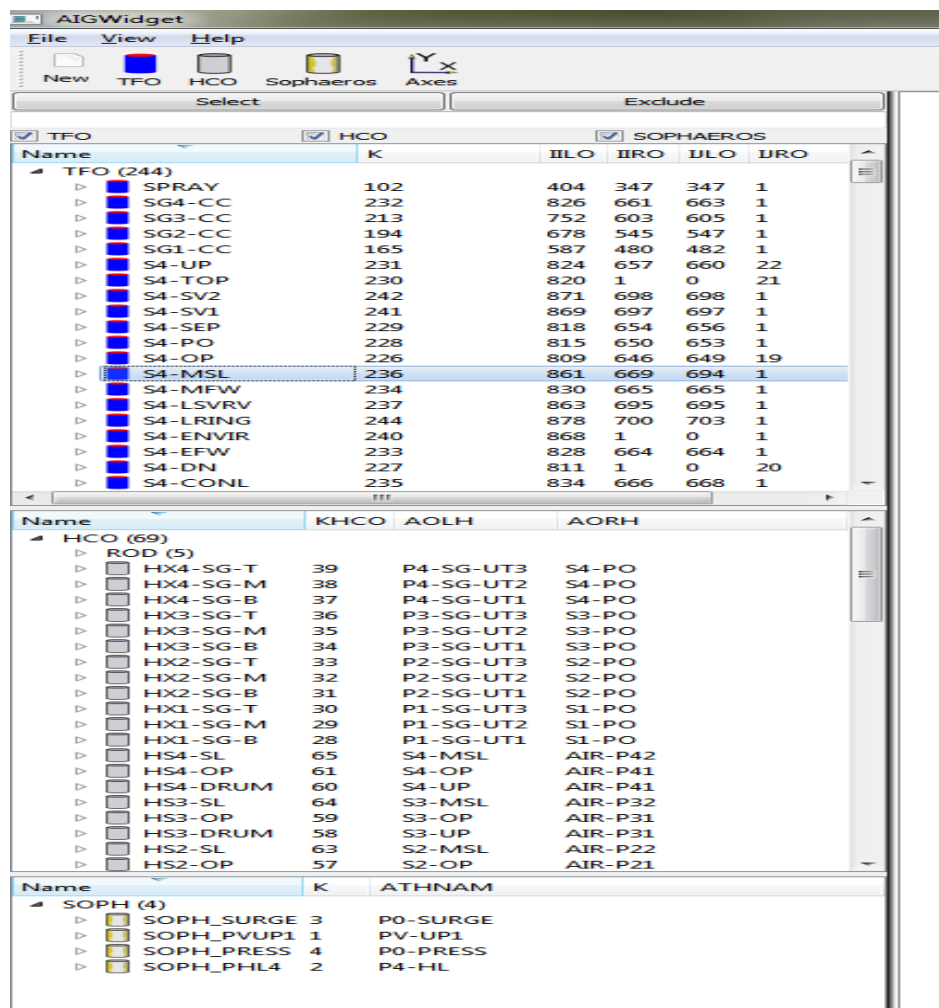


Abb. 4.75 Neue Baumdarstellung der ATHLET-Objekte

Dialogfenster

Des Weiteren sind die aus dem ursprünglichen AIG-Programm bekannten Dialogfenster für die Konfiguration der einzelnen ATHLET-Objekte bzw. der grafischen Szene auf Basis der Qt-Bibliothek neu erstellt worden. Diese Dialogfenster wurden im folgenden Arbeitsschritt mit den Funktionalitäten des auf FORTRAN basierenden Kernprogramms (.dll) verknüpft, um die bereitgestellten Basisfunktionen zu ermöglichen.

Im Einzelnen wurden hierbei die drei Eingabe-Dialoge für die allgemeinen TFO-, HCO und SOPHAEROS-Einstellungen bzw. die drei objektspezifischen Eingabedialoge und der Dialog für die Achsen komplettiert und mit den entsprechenden Eingabeparametern der in Abschnitt 4.3.1 erwähnten FORTRAN-DLL verknüpft. Die neuen Dialoge werden im Abschnitt 4.3.4 durch die Abbildungen Abb. 4.76, Abb. 4.77 und Abb. 4.78 dargestellt.

4.3.4 Verbesserung der Bedienung

Die objektspezifischen Dialoge (s. o.) wurden derart gestaltet, dass ein wiederholtes Schließen der Dialoge nach Beendigung der jeweiligen grafischen Objektbearbeitung überflüssig wird.

Des Weiteren wurden die Interaktionsmöglichkeiten für den Nutzer dahingehend verbessert, dass gewisse Funktionen, wie z. B. der grafische Zoom, direkt über die grafische Oberfläche angesprochen (Mouse Wheel) werden können, ohne dass hierfür jeweils der Umweg über Dialogfenster genommen werden muss (altes AIG-Programm). Eine weitere Verbesserung der Bedienbarkeit wurde durch die Berücksichtigung der vorliegenden Objektverknüpfungen innerhalb der erstellten Objektbaumstruktur realisiert, d. h. miteinander verbundene Objekte (TFO, SOPHAEROS, HEATCOND, RODS) werden in den individuellen Objekt-Baumstrukturen kenntlich gemacht und aufgelistet.

Achsenparalleles Verschieben der Objekte

Zur weiteren Unterstützung der nutzerseitigen grafischen Interaktion im Zusammenhang mit der neu implementierten Drag-and-Drop-Möglichkeit der verschiedenen Objekte wurde die Funktionalität des achsenparallelen Verschiebens mit aufgenommen. D. h. nach der Selektion eines Objektes in der grafischen Oberfläche kann durch Mausbewegung und zusätzliches Betätigen der X- bzw. Y-Taste ein ausschließliches Verschieben der Objekte entlang der X- bzw. Y-Achsen erreicht werden.

APG-Export und Nutzerfreigabe

Das AIG-Programm ist zur Nutzung seit Anfang 2017 freigegeben, die hierbei aufgetretenen nutzerseitigen Verbesserungsvorschläge wurden aufgenommen und noch bestehende Fehler in der Funtionalität behoben. Als zusätzliche Funktion wurde die Möglichkeit geschaffen, die erstellten Bilder der ATHLET Input Graphic in das ATLAS APG-Format zu exportieren. Eine Gesamtdarstellung der erstellten Applikation ist in Abb. 4.79 zu finden.

Gewährleistung der Restartfähigkeit

Für die Gewährleistung der Restartfähigkeit wurde zusätzlich eine C++-seitige Leseroutine für die von der FORTRAN-DLL erzeugte Design-Datei (.ds) erstellt, um die in dieser Datei enthaltenen nutzerabhängigen Modifikationen der individuellen Grafikerstellung bei Neustart des Programms zu reproduzieren und zu erfassen.

Diese Datei wird zwar bereits von der FORTRAN-DLL gelesen, jedoch stellt die vorliegende AthlgrfGetOption-Schnittstellenroutine nicht alle Information für ein verlustfreies Aufsetzen auf dem ursprünglichen Grafik-Status bereit, sodass ein Erstellen der C++-seitigen Leseroutine notwendig wurde.

Dokumentation des erstellten Quellcodes

Der erstellte C++-Quellcode wurde unter Verwendung des Dokumentationswerkzeuges Doxygen hinsichtlich der verwendeten Klassen und deren Methoden eingehend dokumentiert /KON 17/.

Dialogmasken und GUI von AIG

Die folgenden Abbildungen zeigen die neu erstellten Dialogmasken und die grafische Bedienoberfläche anhand von Beispieldaten einer ATHLET-Simulationsanwendung.

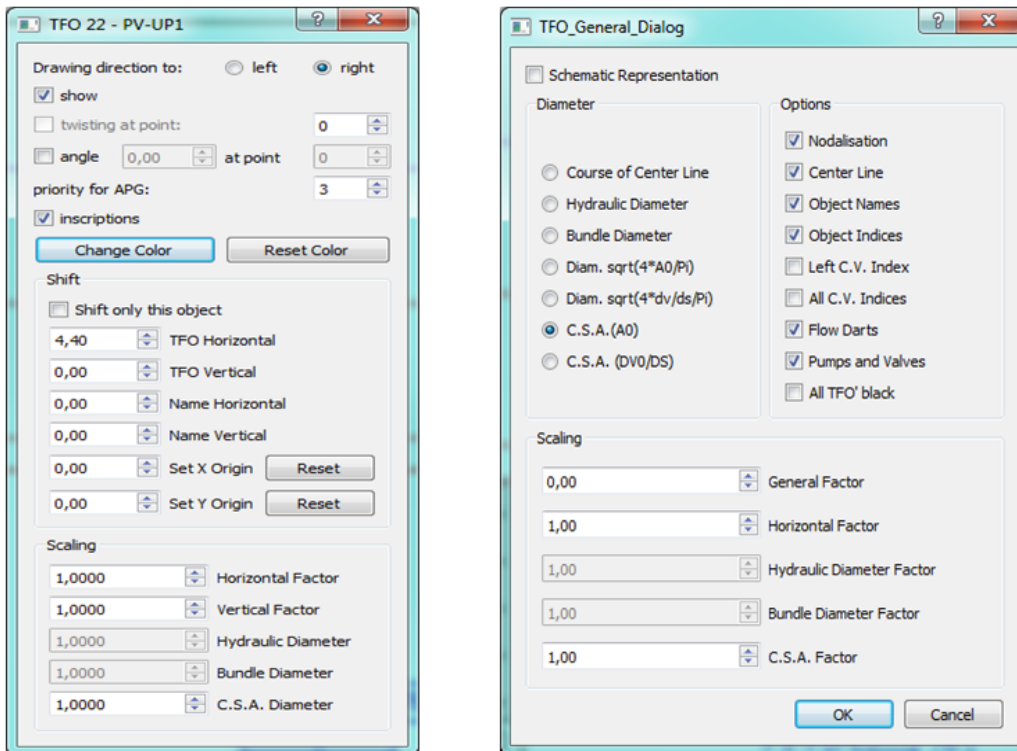


Abb. 4.76 Spezifischer (links) und genereller Dialog (rechts) der TFO-Objekte

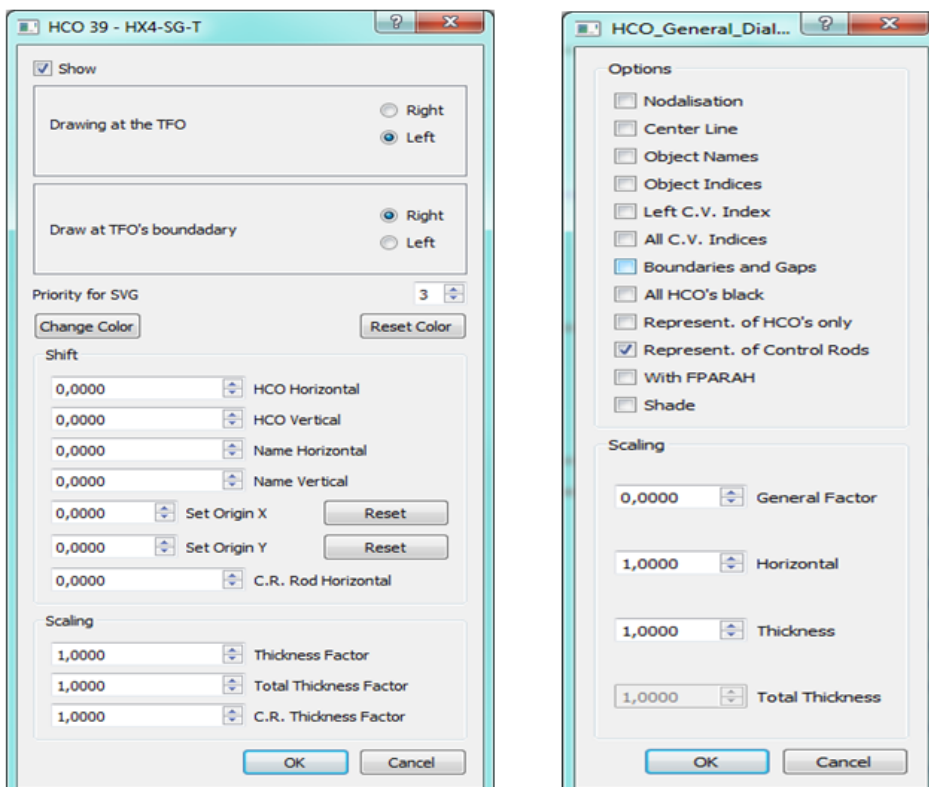


Abb. 4.77 Spezifischer (links) und genereller Dialog (rechts) der HCO-Objekte

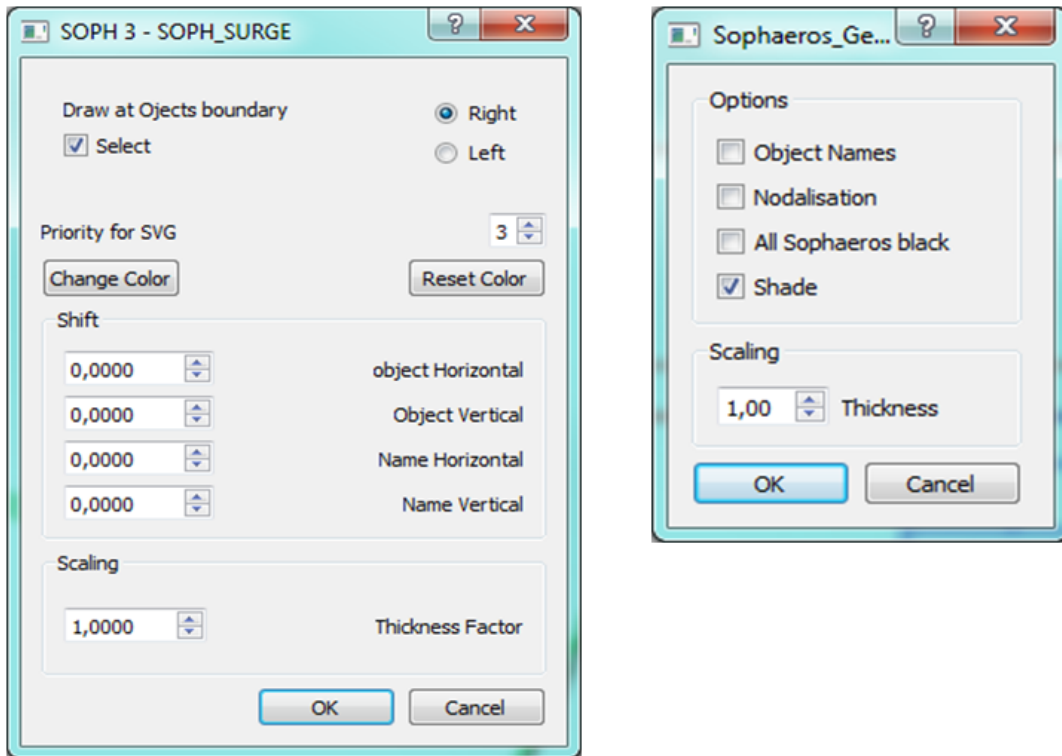


Abb. 4.78 Spezifischer (links) und genereller Dialog (rechts) der SOPHAEROS-Objekte

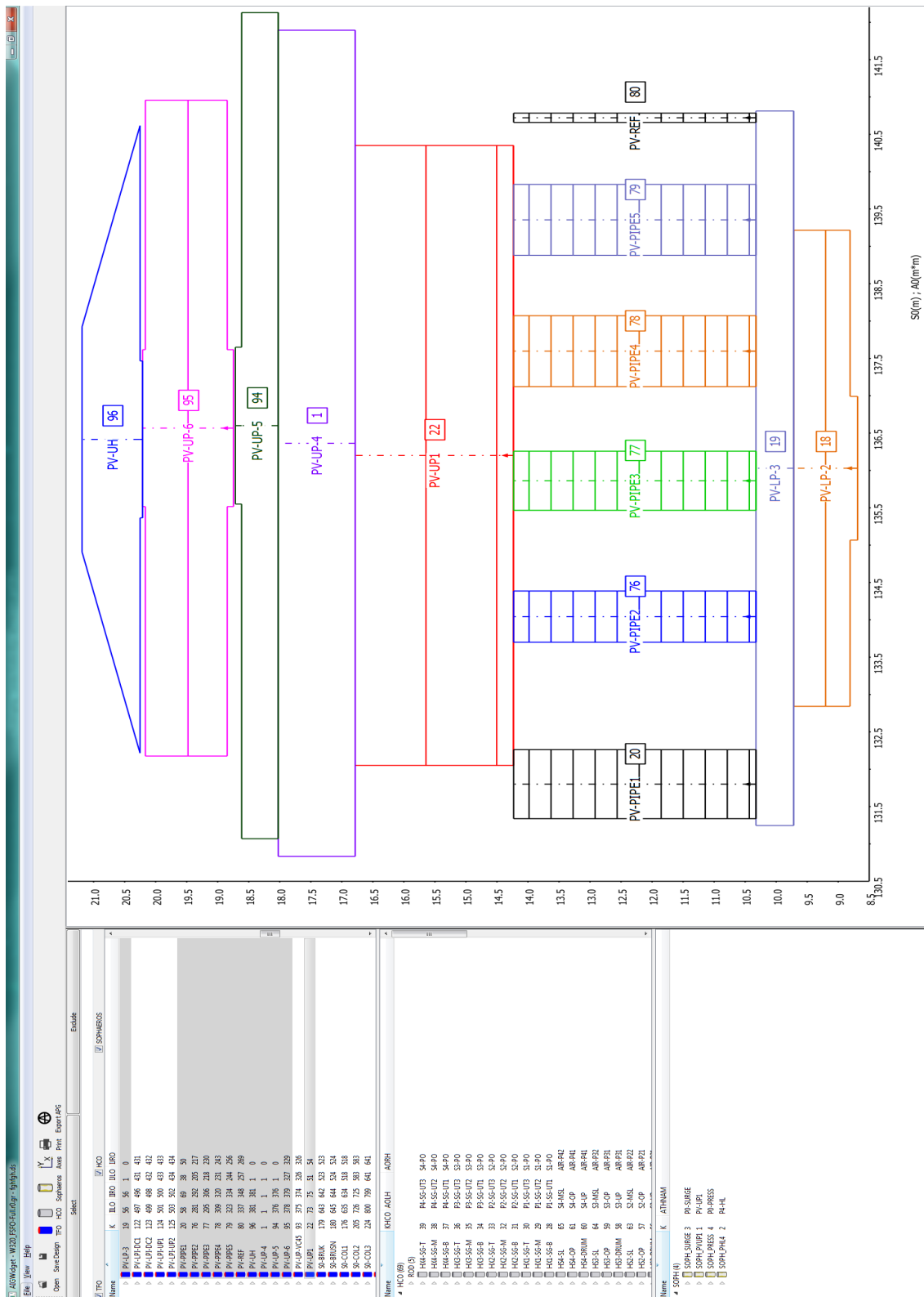


Abb. 4.79 Die neue ATHLET Input Graphik auf Basis der Qt-Bibliothek

5 Zusammenfassung und Ausblick

Das wesentliche Ziel der Arbeiten war, die vorhandene generelle Plattform ATLAS-GRAMOVIS für den Systemcode ATHLET so weiter zu entwickeln, dass sie alle Schritte der Simulation

- Erstellung der Eingabedaten für die Modelle
- Simulationsdurchführung
- Ergebnisauswertung und Visualisierung

mit zeitgemäßen Methoden unterstützt. Die dafür entwickelten Werkzeuge haben einen Stand erreicht, der sie bereits für die Anwendung in der Praxis einsetzbar macht. Es besteht aber in Teilbereichen noch Bedarf für Erweiterungen und Verbesserungen, die in den folgenden Abschnitten beschrieben sind. Neben einer Umsetzung dieser Punkte sollen auch künftig die vorhandenen Werkzeuge in ATLAS-GRAMOVIS für weitere Codes aus AC², insbesondere ATHLET-CD und COCOSYS, ausgebaut werden.

5.1 ATHLET Input Modeller

Mit dem erreichten Entwicklungsstand ist der ATHLET Thermohydraulic Modeller ATM in der Lage, die interaktive Erstellung eines wichtigen Teils der ATHLET Eingabe zu unterstützen. Mit den parametrischen Modellierungsansätzen können die erstellten Daten einfach und effizient angepasst werden, beispielsweise für Sensitivitätsuntersuchungen. Tests an Beispieldatensätzen von ATHLET haben die praktische Einsetzbarkeit gezeigt. Weitere Anwendungen bei der Erstellung detaillierter Anlagenmodelle sollten dennoch ergänzt werden.

Der große, fortlaufend erweiterte Modellumfang mit den umfangreichen Eingabedaten kann dennoch von ATM derzeit noch nicht vollständig abgedeckt werden. Es ist außerdem zu erwarten, dass sich aus der weiteren praktischen Anwendung Probleme ergeben und dadurch Änderungs- und Erweiterungsbedarf aufgezeigt wird. Insbesondere die Importfähigkeit bestehender ATHLET Datensätze muss noch ausgebaut werden, um den Einsatz von ATM auch für diese Daten zu erleichtern.

Der ATHLET GCSM Modeller AGM wird schon seit längerer Zeit zur Modellierung von realen Anlagensystemen praktisch eingesetzt. Im Vorhaben wurden Verbesserungen in

der Handhabung, der Dokumentation und den Schnittstellen zur Basissoftware SimIn-Tech durchgeführt.

Mit der Bereitstellung einer Linux-Version von AGM ist ein Schritt in Richtung Plattformunabhängigkeit durch die Portierung auf FreePascal gelungen. Allerdings sind die Entwicklungswerkzeuge noch nicht auf dem gleichen Stand wie die Delphi-Entwicklungsumgebung RAD-Studio. Daher wird diese gegenwärtig noch zu Weiterentwicklung verwendet. Mittelfristig ist aber zu erwarten, dass ein genereller Wechsel durchführbar ist und dann die Delphi Version des Modellierers nicht mehr benötigt wird.

Mit dem Werkzeug zur Konvertierung vorhandener GSCM-Modelle aus G2 können diese bei Bedarf für AGM aufbereitet und importiert werden, so dass die Weiterarbeit in diesem Werkzeug erfolgen kann. Mittelfristig ist daher zu erwarten, dass AGM das alte Werkzeug G2 vollständig ersetzt.

Insgesamt kann die AGM Entwicklung als ausgereift bezeichnet werden. Allerdings sind Erweiterungen im Hinblick auf neue Simulationsblöcke zur Nachbildung digitaler Leittechnik nötig, die mittlerweile in GCSM verfügbar sind.

5.2 ATLASneo

Eine weitere Zielsetzung war das Reengineering und die Teilerneuerung von ATLAS. Die Entwicklungsprioritäten wurden dazu auch anhand einer Anwenderumfrage zu den Anwendungsfällen gespiegelt. Die klaren Anforderungen die Handhabung und Einsetzbarkeit der Anwendung zu verbessern sowie deren Benutzeroberfläche allgemein zu modernisieren führten dazu, die Gesamtanwendung ATLAS zu modularisieren und als ATLASneo auf Basis aktueller Softwaretechnologien neu zu entwickeln. Hierzu wurde die Rahmenanwendung in Python neu implementiert und die funktionalen Bestandteile der Anwendung unter Einbindung frei verfügbarer Standardpackages als Funktionsmodule neu aufgebaut. Für die Entwicklung dieser, für Datenzugriff und Visualisierung zuständigen Module, wurde großer Wert auf die Verwendung standardisierter Datenformate gelegt. Als zentrales Datenformat für die Speicherung von Simulationsdaten wird zukünftig auf das für große Datenmengen ausgelegte Format HDF5 gesetzt. Das APG-Format zur Beschreibung graphischer Szenen für die schematische Visualisierung wurde durch das SVG-Format abgelöst. Zur Umwandlung alter APG-Bilder wurde ein Konvertierungsprogramm erstellt.

Sowohl die Rahmenanwendung von ATLASneo als auch die im Abschnitt 4.2.3 beschriebenen Funktionsmodule haben das Beta-Stadium erreicht und finden durch interne Anwender erste Verwendung. Die dadurch erhaltenen Rückmeldungen konnten bereits zu Verbesserungen führen oder werden in zukünftigen Weiterentwicklungen miteingeplant.

Die bisherigen Entwicklungen sowie der aktuelle Funktionsumfang zur Ansteuerung und Visualisierung in ATLASneo sind auf die Verwendung von ATHLET ausgerichtet. Besonders in den Funktionsmodulen zur Online-Simulation und schematischen Visualisierung wird dies deutlich. Da die zu Grunde gelegten Konzepte aber sehr viel allgemeiner anwendbar sind, sollten diese in ihrer weiteren Entwicklung verallgemeinert und erweitert werden, um auch andere Rechencodes unterstützen zu können.

5.3 ATHLET Input Graphic

Die durchgeführten Reengineering-Aufgabe für das Post-Process Programm AIG konnten erfolgreich bearbeitet werden. Hierbei konnten alle wesentlichen Funktionen des AIG-Vorläuferprogramms in die neue AIG-Oberfläche übernommen und erweitert werden. Der festgestellte Optimierungsbedarf bei den handgeführten Eingabegeräten (Mouse, Trackball, etc.) konnte in der überarbeiteten AIG-Version durch verbesserte Anzeige- und Suchfunktion, Oberflächen-Handhabbarkeit für die verschiedenen graphischen Objekte und durch erweiterte Nutzerinteraktionsmöglichkeit wesentlich verringert werden. Der aktuell erreichte Stand des AIG-Programms ermöglicht durch die neu erstellten Strukturen ebenfalls bestehende und zukünftig entstehende Nutzeranforderungen mit überschaubarem Aufwand zu implementieren.

Literaturverzeichnis

- /APT 12/ Applied Programming Technology, Inc., Symbolic Nuclear Analysis Package (SNAP), User's Manual, October 2012.
- /AUS 12/ Austregesilo, H. et al., ATHLET Mod 3.0 Cycle A, Models and Methods, GRS-P-1/Vol. 4, Rev. 3., 2012.
- /BLI 17/ Blink – Open Source Rendering Engine used by Chromium,
Bearbeitungsstand: 7. Juli 2017, 13h00, URL: <https://chromium.org/blink>.
- /EMB 15/ Embarcadero Technologies, Delphi XE 7,
Bearbeitungsstand: 15. Februar 2015, 13h00,
URL: <http://www.embarcadero.com/de/products/delphi>.
- /FAH 18/ FastHelp is a Windows Help File Generator,
Bearbeitungsstand: 05. Dezember 2018,
URL: <https://www.fast-help.com/>.
- /FPC 15/ Free Pascal - Open Source Compiler for Pascal and Object Pascal,
Bearbeitungsstand: 15. Juni 2017, 13h00,
URL: <https://www.freepascal.org/>.
- /GIT 18/ Git is a free and open source distributed version control system,
Bearbeitungsstand: 05. Dezember 2018, 13h00,
URL: <https://git-scm.com/>.
- /HDF 17/ HDF5, a technology suite that makes possible the management of extremely large and complex data collections,
Bearbeitungsstand: 10. Juli 2017, 09h45,
URL: <https://support.hdfgroup.org/HDF5/>.
- /IDE 66/ Idel'chik, I.E., Handbook of Hydraulic Resistance, Israel Program for Scientific Translations Ltd. IPST Cat. No. 1505, 1966.

- /INK 17/ Inkscape – Open Source Vector Graphics Editor using SVG as the native format, Bearbeitungsstand: 7. Juli 2017, 13h00, URL: <https://inkscape.org/>.
- /JAK 97/ Jakubowski, Z., Leittechnikmodule des Reaktorschutz-Reaktorleistungsbegrenzungs-systems und der elektrischen Stromversorgung für das KKW Neckar II (GKN II), GRS-A-2542, 1997.
- /KON 17/ Technische Notiz zur AIG-Code Dokumentation, Version 1, 27.06.2017.
- /LER 16/ Lerchl, G.; et al., ATHLET Mod 3.1 Cycle A User's Manual, GRS-P-1/Vol. 1 Rev. 7, 2016.
- /MFC 14/ Microsoft Developer Network, MFC Desktop Applications, Bearbeitungsstand: 10. Juli 2014, 13h00, URL: <http://msdn.microsoft.com/en-us/library/d06h2x6e.aspx>
- /NIM 17/ Nim – Systems and applications programming language, Bearbeitungsstand: 10. Juli 2017, 08h00, URL: <https://nim-lang.org/>.
- /OGL 14/ OpenGL, The Industry's Foundation for High Performance Graphics, Bearbeitungsstand: 10. Juli 2014, 13h00, URL: <http://www.opengl.org>
- /PYQ 17/ PyQtGraph, Scientific Graphics and GUI Library for Python, Bearbeitungsstand: 10. Juli 2017, 09h00, URL: <http://www.pyqtgraph.org/>.
- /PYS 15/ PySide Python for Qt, provides LGPL-licensed Python bindings for Qt, Bearbeitungsstand: 10. Juli 2017, 09h20, URL: <https://wiki.qt.io/PySide>.
- /PYT 17/ Python Software Foundation, Bearbeitungsstand: 7. Juli 2017, 13h00, URL: <https://www.python.org/>.
- /QTC 17/ The Qt Company, C++-Bibliothek, Bearbeitungsstand: 7. Juli 2017, 13h00, URL: <https://www.qt.io/#>.

- /QWE 17/ QWebView, a widget provided by WebKit in Qt used to view and edit web documents, Bearbeitungsstand: 10. Juli 2017, 09h30, URL: https://wiki.qt.io/Open_Web_Page_in_QWebView.
- /SCA 11/ Scalable Vector Graphics (SVG) 1.1 (Second Edition), Bearbeitungsstand: 7. Juli 2017, 13h00, URL: <https://www.w3.org/TR/SVG11/>.
- /SIT 15/ 3V Services, SimInTech, Bearbeitungsstand: 10. Februar 2015, 13h00, URL: http://3vservices.com/ru/index.php?option=com_content&view=article&id=68&Itemid=69.
- /TRA 18/ Trac Open Source Project, Bearbeitungsstand: 05. Dezember 2018, 13h00, URL: <http://trac.edgewall.org/>.
- /VDI 84/ VDI Wärmeatlas, 4. Auflage 1984.
- /VOG 15/ Voggenberger, T. et al., April 2015, Entwicklung neuer Methoden zur Modellierung technischer Systeme und zur Ergebnisauswertung für Simulationsprogramme der Reaktorsicherheit, GRS-368.
- /VOG 18/ Voggenberger, T., 2018, ATHLET-Input-Modeller User Manual.
- /WEB 91/ WebKit – Open Source Web Browser Engine, Bearbeitungsstand: 7. Juli 2017, 13h00, URL: <https://webkit.org/>.
- /WIK 17/ Wikipedia, SharePoint, Bearbeitungsstand: 15. Juni 2017, 13h00, URL: <https://de.wikipedia.org/wiki/SharePoint>.
- /WIK 17a/ Wikipedia, Javascript (kurz JS), Bearbeitungsstand: 7. Juli 2017, 13h00, URL: <https://de.wikipedia.org/wiki/JavaScript>.

Abbildungsverzeichnis

Abb. 4.1	Menüleiste in ATM zur Erstellung von TFO.....	13
Abb. 4.2	Gegenwärtig verfügbare Fluidobjekte	13
Abb. 4.3	Geometrische Basiselemente für ein TFO	14
Abb. 4.4	Schematische und geometrische Darstellung eines TFO in ATM.....	14
Abb. 4.5	Vorgabe der Anfangskordinaten eines TFO	15
Abb. 4.6	Verfügbare Komponenten für ein TFO	15
Abb. 4.7	Eigenschaften für ein Rohrstück (Property-Fenster)	16
Abb. 4.8	Einfügen und Verbinden von hydraulischen Basiselementen.....	17
Abb. 4.9	Geometriedarstellung nach Verbindung der Basiselemente.....	17
Abb. 4.10	Datenänderung in den Basiselementen des TFO.....	18
Abb. 4.11	Eigenschaften für eine Komponente (Pumpe).....	18
Abb. 4.12	Automatische Positionierung von Komponenten.....	19
Abb. 4.13	ATHLET-spezifisches Menü in ATM	20
Abb. 4.14	Darstellung von Komponenten in der schematischen Ansicht.....	20
Abb. 4.15	Anzeige von ortsabhängigen Daten im TFO table editor	21
Abb. 4.16	Vorgabe von „Branching“ und „Branch2M“ Daten im TFO Table Editor	22
Abb. 4.17	Beispiel für ein TFO mit detaillierter Diskretisierung.....	24
Abb. 4.18	Formverlustkoeffizienten für einen Diffusor mit verschiedenen Auswahlmöglichkeiten	25
Abb. 4.19	Modellierung eines Fluidkreislaufs in ATM.....	26
Abb. 4.20	Modell einer Leitung (Surgeline) im Fluidsystem.....	27
Abb. 4.21	Zuordnung von TFO zu Prioritätsketten	28
Abb. 4.22	Kopplung zweier TFOs mit einem HCO	29
Abb. 4.23	Property Dialog für ein Wärmeleitobjekt.....	30
Abb. 4.24	Ein CCO verbindet zwei parallele Pipes.....	31
Abb. 4.25	Alte (links) und neue (rechts) Editoren für ein CCO	32

Abb. 4.26	Ad-hoc Generierung des ATHLET Codes für ein CCO	33
Abb. 4.27	Datenspezifikation für Eingabedatei und Simulation	34
Abb. 4.28	ATHLET Ausgabe im Standardeditor von ATM.....	35
Abb. 4.29	TFO Daten in ATHLET ASCII Datei	37
Abb. 4.30	ATHLET Objekttopologie in GR-Ausgabedatei	37
Abb. 4.31	Automatisch importierte TFO in ATM.....	38
Abb. 4.32	Beispiel des Datenimports für ein CCO.....	39
Abb. 4.33	Eingabefelder für ASCII-Datenimport (Simulation properties)	39
Abb. 4.34	Beispiel für Eingabedaten im ATM zur ATHLET Steuerung	40
Abb. 4.35	Daten für Simulationsausführung (Simulation properties)	41
Abb. 4.36	Hinweise im „Message Window“ beim Erzeugen der Eingabedaten	42
Abb. 4.37	Workspace für die Definition von Parametern	43
Abb. 4.38	Verwendung von Parametern im Property Dialog	43
Abb. 4.39	Ausgabe der Parameter Definitionen im ATHLET Format.....	44
Abb. 4.40	TFO als Kopie eines Masterobjekts	45
Abb. 4.41	Kopie eines TFO in ATHLET	45
Abb. 4.42	Makro-Objekt für Parallelkanal-Modellierung eines Zylinders.....	46
Abb. 4.43	Simulation eines vertikalen Zylinders mit parallelen TFO	47
Abb. 4.44	Export eines Workspaces aus ATM ins SVG-Format.....	48
Abb. 4.45	Primärkreislaufdarstellung als SVG-Export aus ATM.....	49
Abb. 4.46	ATM Workspace mit dynamischen Daten in ATLASneo.....	50
Abb. 4.47	Anbindung der interaktiven Hilfe in ATM.....	52
Abb. 4.48	Beispielseite für interaktive Hilfe in ATM.....	53
Abb. 4.49	Linux und Windows Version des „About“-Fensters.....	55
Abb. 4.50	Systemschaltbild einer 2-Punkt-Regelung	56
Abb. 4.51	Simulation einer 2-Punkt-Regelung (Linux und Windows).....	57
Abb. 4.52	Blockdaten in der Exportdatei des G2-Modellierers	58

Abb. 4.53	Arbeitsblatt eines Systemmodells im G2-Modellierer	59
Abb. 4.54	Konvertiertes Arbeitsblatt eines Systemmodells in AGM.....	60
Abb. 4.55	Anwendungsbeispiel „Containment“	70
Abb. 4.56	Linien und Flächeneigenschaften	70
Abb. 4.57	Einstellwinkel und Kreise	71
Abb. 4.58	Vektoren	71
Abb. 4.59	Mehrfachstart und Titel von Funktionsmodulen.....	72
Abb. 4.60	Drag-and-Drop in Aktion: der Maus-Zeiger signalisiert durch seine Form, ob eine Verknüpfung möglich ist (links) oder nicht (rechts).....	75
Abb. 4.61	Drag-and-Drop ungeeigneter Daten: Der Tooltip am Maus-Zeiger zeigt dem Benutzer welche Datenelemente ignoriert werden und warum.....	76
Abb. 4.62	Das HDF5Panel kann HDF5-Dateien einladen und stellt deren Inhalt hierarchisch dar – der Bereich unten zeigt die enthaltenen Metainformationen.	78
Abb. 4.63	Steuermodul für Online ATHLET-Simulationen: Zusätzlich zu Steuerelementen werden der aktuelle Simulationszustand (State Data) sowie Plotdaten (Plot Data) in Form eines Baumes dargestellt.	79
Abb. 4.64	Das Plotmodul im Einsatz: Die zu plottenden Größen wurden im Plotdatenbaum einer Online-Simulation (MonitorWidget) sowie einer HDF5-Datei (HDF5Panel) ausgewählt und per Drag-and-Drop den Plotfenstern zugeteilt.	82
Abb. 4.65	Farbbalken mit Kontextmenü	86
Abb. 4.66	Variablenauswahl für dynamisches Textobjekt	87
Abb. 4.67	ATHLET "sample1" in EIDOS	88
Abb. 4.68	Online-Simulation in ATLASneo: Verwendung des Funktionsmoduls GCSMViz und des Plotmoduls zur Analyse der Leittechnik sowie des zeitlichen Verlaufs von Zustandsvariablen	90
Abb. 4.69	Ursprüngliches Konzept für die neue AIG-Oberfläche.....	92
Abb. 4.70	Überarbeitetes Konzept für die neue AIG-Oberfläche	92
Abb. 4.71	Begrenzungsrahmen ermittelt durch die Qt-Klasse SharedRenderer	94
Abb. 4.72	Verbesserte Methode zur Bestimmung des Begrenzungsrahmens.....	94

Abb. 4.73	Verbessertes Konzept der FORTRAN-DLL-AIG-Einbindung.....	95
Abb. 4.74	Darstellung der TFO, HCO und SOPAHEROS-Objekte der alten AIG-Version	96
Abb. 4.75	Neue Baumdarstellung der ATHLET-Objekte	97
Abb. 4.76	Spezifischer (links) und genereller Dialog (rechts) der TFO-Objekte.....	100
Abb. 4.77	Spezifischer (links) und genereller Dialog (rechts) der HCO-Objekte	100
Abb. 4.78	Spezifischer (links) und genereller Dialog (rechts) der SOPHAEROS- Objekte	101
Abb. A.1	Abmessungen des Diffusors.....	122
Abb. B.1	Aufruf des Konvertierungsprogramms	125
Abb. B.2	Aufruf des Sortierungsprogramms.....	126
Abb. C.1	Nutzen Sie manuell erstellte Prozessbilder mit speziellen Funktionen?...	127
Abb. C.2	Nutzen Sie Steuerungsmöglichkeiten?	127
Abb. C.3	Nutzen Sie automatisch erzeugbare Bilder?	128
Abb. C.4	Passen Sie X/Y-Diagramme an?	128

Tabellenverzeichnis

Tab. 4.1	ATLAS Anwendungsfälle (use cases).....	62
Tab. 4.2	Erweiterte MIME-types, welche bisher in ATLASneo verwendet werden.....	76

Abkürzungsverzeichnis

AGM	ATHLET GCSM Modeller
AIG	ATHLET Input Graphics
APG	ATLAS Picture Generator
ATHLET	Analyse der Thermohydraulik von Lecks und Transienten
ATLAS	Athlet Analyse Simulator
ATM	ATHLET Thermohydraulic Modeller
CSS	cascading stylesheets
DLL	dynamische Bibliothek
DOM	Document Object Model
GRAMOVIS	grafische Modellierung und Visualisierung
GUI	Bedienoberflächenelement
HCO	Wärmeleitungsobjekte (Heat conduction objects)
PCO	“Priority-Chain“-Objekte
RDB	Reaktordruckbehälter
SGMOT	Signal für Motorschaltung
SGPUMP	Signal für Pumpendrehzahl
SVG	Scalable Vector Graphics
TFO	Thermo-Fluidobjekte

A Anhang: Berechnung von Formverlustkoeffizienten in ATM

Alle Berechnungen erfolgen auf Basis von empirischen Grundlagen aus Idel'Chik /IDE 66/ oder dem VDI-Wärmeatlas /VDI 84/. Für Idel'Chik wurden die Zetawerte entsprechend der Kurven mit Funktionen approximiert, die Werte des VDI Wärmeatlas sind tabellarisch vorgegeben und werden interpoliert. Alle Werte sind für glatte Rohrstücke mit hohen Reynoldszahlen. Der Reibungsanteil von Zeta wird für $RD > 1$ in ATHLET zusätzlich berücksichtigt.

A.1 Rohrbogen

D: Innendurchmesser des Rohrbogens

R: Radius des Rohrbogens

δ : Winkel des Rohrbogens (15 ... 180 grad)

A: Empirischer Wert

$$A = -1.2371 - 0.49833 \cdot \ln(\delta) \quad (15 \leq \delta < 180 \text{ grad})$$

$$A = 0.0 \quad (\delta < 15 \text{ grad})$$

λ : Darcy-Weissbach friction factor

(ist von der Reynoldszahl abhängig und wird fest mit 0.02 berücksichtigt)

$$RD = R/D$$

A.1.1 Idel'Chik

Für $RD < 1.0$:

$$ZETA = \zeta = \frac{\pi}{180} \cdot \lambda \cdot RD \cdot \delta + A \cdot 0.21 / RD^{2.5} \quad \text{für } 15 \leq \delta < 180 \text{ grad}$$

$$ZETA = \zeta = \frac{\pi}{180} \cdot \lambda \cdot RD \cdot \delta \quad \text{für } \delta < 15 \text{ grad}$$

Für $RD \geq 1.0$:

$$ZETA = \zeta = A \cdot 0.21 / RD^{0.5} \quad \text{für } 15 \leq \delta < 180 \text{ grad}$$

$$ZETA = \zeta = 0 \quad \text{für } \delta < 15 \text{ grad (wird in ATHLET bereits intern berücksichtigt)}$$

Formverlustkoeffizienten für ATHLET:

$$ZFFJ0 = ZFFB0 = ZETA / (D^2 \cdot \pi / 4)^2$$

A.1.2 VDI-Wärmeatlas

Für $RD < 1.0$:

Keine Werte vorhanden! >> es werden die Werte gemäß Idelchik verwendet!

Für $RD \geq 1.0$:

RD / δ	15	30	45	90	180
1	0.03	0.07	0.15	0.2	0.3
2	0.03	0.06	0.09	0.145	0.15
3	0.03	0.06	0.08	0.12	0.14
4	0.03	0.06	0.075	0.11	0.15
5	0.03	0.06	0.07	0.10	0.13
6	0.03	0.06	0.07	0.09	0.12
7	0.03	0.06	0.07	0.09	0.12
8	0.03	0.06	0.07	0.10	0.13
9	0.03	0.06	0.07	0.10	0.14
10	0.03	0.06	0.07	0.11	0.15

A.2 Kniestück

Die Formeln und die Implementierung berücksichtigen verschiedene Winkel. Momentan wird in ATM nur ein Kniestück mit 90 Grad (ATH_BEND) verwendet. Bei den anderen Bögen wird unabhängig von der definierten Anzahl der Segmente immer ein idealer Rohrbogen für die Formverluste angenommen.

A.2.1 Idel'Chik

D: Innendurchmesser des Rohrbogens

R: Radius des Rohrbogens

δ : Winkel des Knicks (15 ... 180 grad)

λ : Darcy-Weissbach friction factor

(ist von der Reynoldszahl abhängig und wird fest mit 0.02 berücksichtigt)

$$RD = R/D$$

$$R0D0 = RD * \cos(\delta/2)$$

$$ZETA1 = 1.0 / (-0.4425 + 2.8 * R0D0 - 0.4075 * R0D0^2 + 0.0178 * R0D0^3)$$

$$L0D0 = 2 * R0D0 * \tan(\delta/2)$$

$$ZETA2 = 2 * \lambda * L0D0$$

$$ZETA = ZETA1 + ZETA2$$

Diese Formel ist in ATHLET programmiert, wurde aber wegen des nicht vorhandenen Segmentbogens nicht verwendet. Im Vergleich zum VDI-Wärmeatlas ergibt sich für einen 90-Grad-Bogen mit $R0D0 = 4$ ein Wert von 0.44, also eine sehr große Abweichung.

Formverlustkoeffizienten für ATHLET:

$$ZFFJ0 = ZFFB0 = ZETA / (D^2 * \pi/4)^2$$

A.2.2 VDI-Wärmeatlas

δ : Winkel des Knies (0 ... 120 grad)

δ	10	20	30	40	50	60	70	80	90	100	110	120
ζ	0.04	0.07	0.10	0.20	0.30	0.47	0.67	0.87	1.13	1.40	1.70	2.00

Zusammengesetzte Kniestücke mit gleichem Winkel δ ergeben kleinere Werte als die Summe der Kniestücke dieses Winkels. Der Wert hängt vom Verhältnis der Länge zum Durchmesser der Einzelstücke ab. Vereinfacht wird pauschal ein Faktor von 0.7 angesetzt, mit dem die Summe der Zetawerte multipliziert wird. Somit würde sich für einen 90-Grad-Segmentbogen mit drei 30-Grad-Segmenten ergeben:

$$ZETA = \zeta = 0.7 * (0.1 + 0.1 + 0.1) = 0.21$$

Formverlustkoeffizienten für ATHLET:

$$ZFFJ0 = ZFFB0 = ZETA / (D^2 * \pi/4)^2$$

A.3 Diffusor

d: Durchmesser des Diffusors am Eintritt

D: Durchmesser des Diffusors am Austritt ($D > d$)

l: Länge des Diffusors

α : Öffnungswinkel des Diffusors in Grad

F: Querschnittsverhältnis

F_{60} : Maximalwert bei $\alpha = 60^\circ$

ZETA = ζ : Formverlust am Diffusoreintritt (auf die kleinere Fläche bezogen)

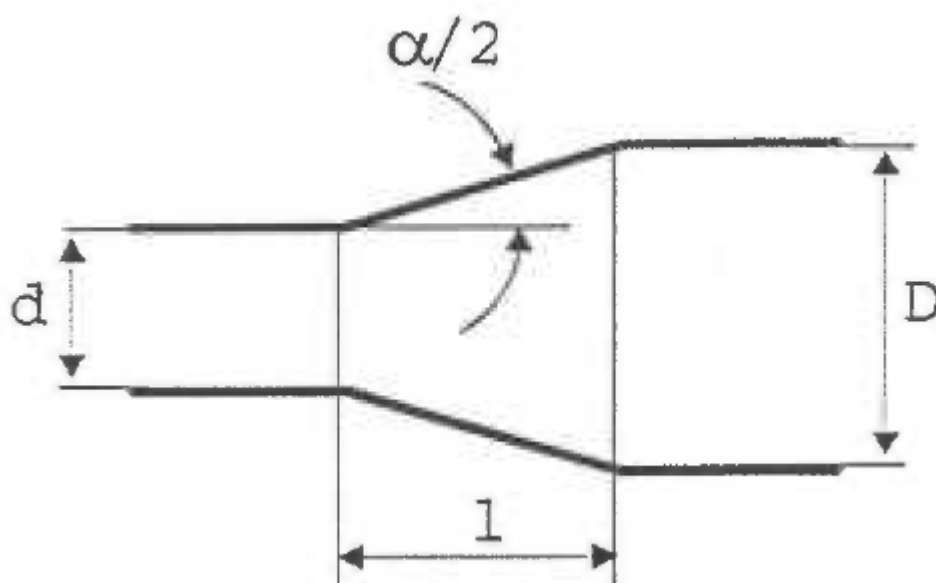


Abb. A.1 Abmessungen des Diffusors

A.3.1 Idel'Chik

$$\alpha = 2 \cdot \arctan \left(\frac{D - d}{2 \cdot l} \right)$$

$$F = F_d / F_D = (d / D)^2$$

$$F_{60} = 1.15 \cdot (1 - F)^2$$

Für $0 < \alpha \leq 60$ Grad:

$$\text{ZETA} = \zeta = F_{60} (\sin(1.5 \alpha))^{1.65}$$

Für $60 < \alpha \leq 180$ Grad:

$$\text{ZETA} = \zeta = F_{60} \cdot (1 - 0.001(\alpha - 60))$$

A.3.2 VDI-Wärmeatlas

Zeta ist abhängig vom Öffnungswinkel und dem Verhältnis der Durchmesser (D/d). Im Diagramm für ζ' gibt es aber nur zwei Kurven für D/d = 1.5 und für D/d = 3.0. Die Unterschiede sind gering und es wird daher der Mittelwert ζ' für alle D/d verwendet.

$$\alpha = 2 \cdot \arctan(D - d) / (2 \cdot l)$$

$$F = F_d / F_D = (d / D)^2$$

$$\text{ZETA} = \zeta = \zeta' \cdot (1 - F)^2 \text{ (Achtung: Ist auf die kleine Fläche (d) bezogen)}$$

Für eine sprunghafte Erweiterung ($\alpha = 180$) ist $\zeta' = 1.0$.

α	0	10	20	30	40	50	60	70	80	90
ζ'	0.20	0.13	0.40	0.70	0.88	1.03	1.10	1.14	1.12	1.08
α	100	110	120	130	140	150	160	180		
ζ'	1.07	1.06	1.05	1.04	1.03	1.02	1.01	1.00		

Formverlustkoeffizienten für ATHLET:

$$\text{ZFFJ0} = \text{ZETA} / (d^2 \cdot \pi / 4)^2$$

ZFFB0 siehe ZFFJ0 bei der Düse!

A.4 Düse (Querschnittsverengung)

Die Düse entspricht einem rückwärts durchströmten Diffusor.

d: Durchmesser der Düse am Austritt

D: Durchmesser der Düse am Eintritt (D>d)

l: Länge der Düse

α : Öffnungswinkel der Düse in Grad

F: Querschnittsverhältnis

ZETA = ζ : Formverlust am Düsenaustritt

(ZFFJ0 wird auf die kleinere Fläche bezogen)

A.4.1 Idel'Chik

Implementiert sind die folgenden Approximationen:

$$\alpha = 2 \cdot \arctan(D - d) / (2 \cdot l)$$

$$F = F_d / F_D = (d / D)^2$$

$$L = l/d - 0.065$$

$$\Psi = 0.11 \cdot L^{-0.15} \cdot \exp(-0.2 \cdot L) \text{ für } L > 0.0$$

$$\Psi = 0.5 - 3.75 \cdot L \text{ für } L \leq 0.0$$

$$\varphi = 120 \cdot (0.01 \cdot \Psi)^{0.125}$$

Für $\alpha \leq \varphi$:

$$\omega = 1 - \alpha / \varphi ; E = 2 + 1.67 \cdot l/d$$

Für $\alpha > \varphi$:

$$\omega = (\alpha - \varphi) / (180 - \varphi) ; E = 1.25$$

Schließlich:

$$\zeta' = \Psi + (0.5 - \Psi) (\omega)^E$$

$$\text{ZETA} = \zeta = \zeta' \cdot (1 - F)$$

A.4.2 VDI-Wärmeatlas

Für eine sprunghafte Verengung ($\alpha = 180$) gilt ($Re = 10^{**4}$):

$$\text{ZETA} = \zeta = 0.52 \cdot (1 - F)$$

Für $\alpha < 40$:

$$\text{ZETA} = \zeta = 0.04$$

Für $\alpha \geq 40$:

$\text{ZETA} = \zeta = 0.5 \cdot (1 - F)$ (in erster Näherung, sollte aber kleiner sein). Hier ist die VDI-Angabe sehr ungenau!

Formverlustkoeffizienten für ATHLET:

$$\text{ZFFJ0} = \text{ZETA} / (d^2 \cdot \text{Pi}/4)^2$$

ZFFB0 siehe ZFFJ0 bei Diffusor!

B Anhang: Zusätzliche Hinweise zur Konvertierung von Leittechnik- modellen

B.1 Verwendung des Konvertierungsprogramms

Das Konvertierungsprogramm kann aus der Konsole wie folgt aufgerufen werden.

```
AGMConverter TEMPLATE INFILE OUTFILE [options]

AGMConverter -h | --help

Converts layout files (.dat) exported by G2 into AGM-format (.xpvt).

Arguments:

TEMPLATE the template file (e.g. template.xpvt). This file should con-
tain prototypes for all translated blocks.

INFILE either an input dat-file exported by G2, or an xpvt-file for
validation and correction of known issues.

OUTFILE the output xpvt-file.

Options:

--checkWires=S switch (0 | 1) to disable/enable validation of wire
types [default: 1]

--sortObjects=S switch (0 | 1) to disable/enable sorting of objects
[default: 1]

--checkLabels=S switch (0 | 1) to disable/enable validation of labels
[default: 0]

-p, --prettify prettify output file

-h, --help show this help screen
```

Abb. B.1 Aufruf des Konvertierungsprogramms

B.2 Abweichungen bei der Konvertierung

- Texte in G2, die keinen Rahmen haben (also nicht mit Elementen aus der GCSM-Palette erzeugt worden sind, sondern mit G2-Systemmitteln), werden in AGM nicht angezeigt. Um diese Texte anzuzeigen, sind sie mit Elementen aus „Palettes-Signal/Comments“ neu zu generieren oder manuell in AGM nachzutragen.
- Bei komplizierten Netzwerken können Konversionsfehler bei den Verbindungen auftreten, die von der Konversionsroutine angezeigt werden („ERROR: ...). Die Verbindungen sind dann von Hand in AGM einzutragen.
- Bei Knoten, die sowohl mit einem Eingangskonnektor als auch mit einem Ausgangskonnektor verbunden sind, können die „To memory“- und „From memory“-Objekte in AGM vertauscht sein. Dies muss in AGM von Hand korrigiert werden.
- Bei Switches kann es in G2 vorkommen, dass ein Eingang nicht belegt ist. In diesem Fall muss dieser Eingang in AGM durch ein Ersatz-GCSM-Objekt belegt werden (z. B. bei einem Analogeingang eine Konstante mit dem Wert 0).

B.3 Verwendung des Sortierprogramms

Das Sortierprogramm kann aus der Konsole wie folgt aufgerufen werden.

```
norm_athlet_input [-h] [-p] [-o <OUTPUT FILE NAME>] <INPUT FILE NAME>

Normalisation of Athlet input files.

positional arguments:
<INPUT FILE NAME>      Path to Athlet input file to be normed.

optional arguments:
-h, --show this help message and exit
-p, --norm-parameter-order

Switch which, if activated, also normalises the parameter order of
some GCSM signals (ADDER, SORT, AND, OR)

-o <OUTPUT FILE NAME>, --output-file <OUTPUT FILE NAME>

Output will be written to the user-defined output file and not to
stdout
```

Abb. B.2 Aufruf des Sortierprogramms

C Anhang: Ergebnisse der Umfrage zu ATLAS Anwendungsfällen

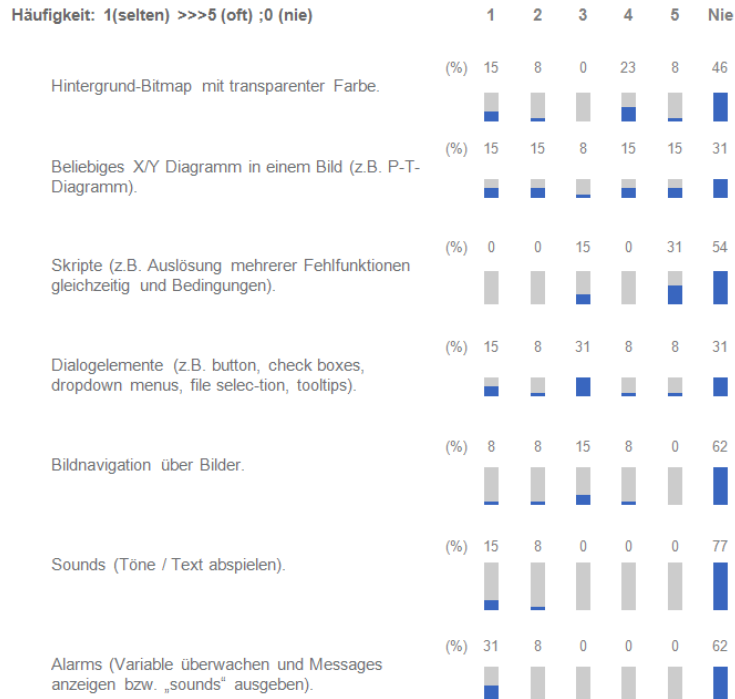


Abb. C.1 Nutzen Sie manuell erstellte Prozessbilder mit speziellen Funktionen?

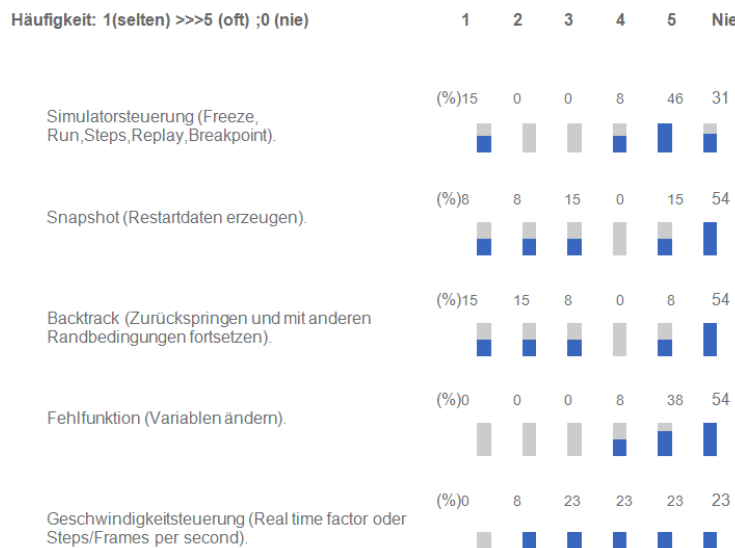


Abb. C.2 Nutzen Sie Steuerungsmöglichkeiten?

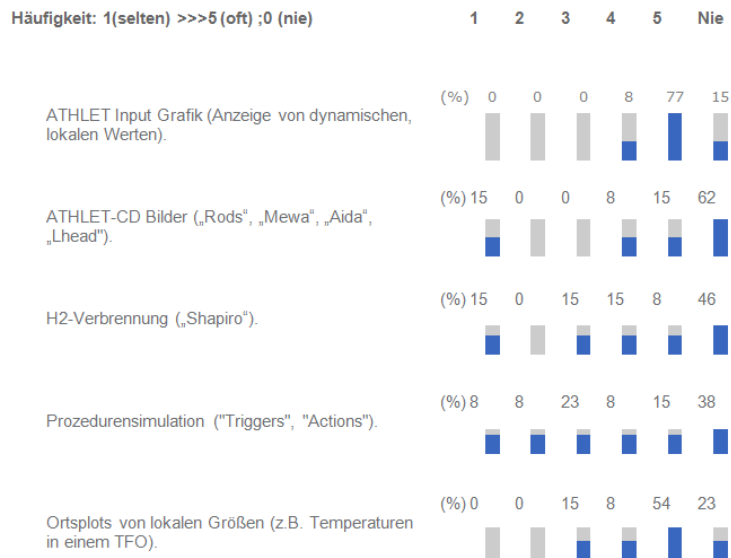


Abb. C.3 Nutzen Sie automatisch erzeugbare Bilder?

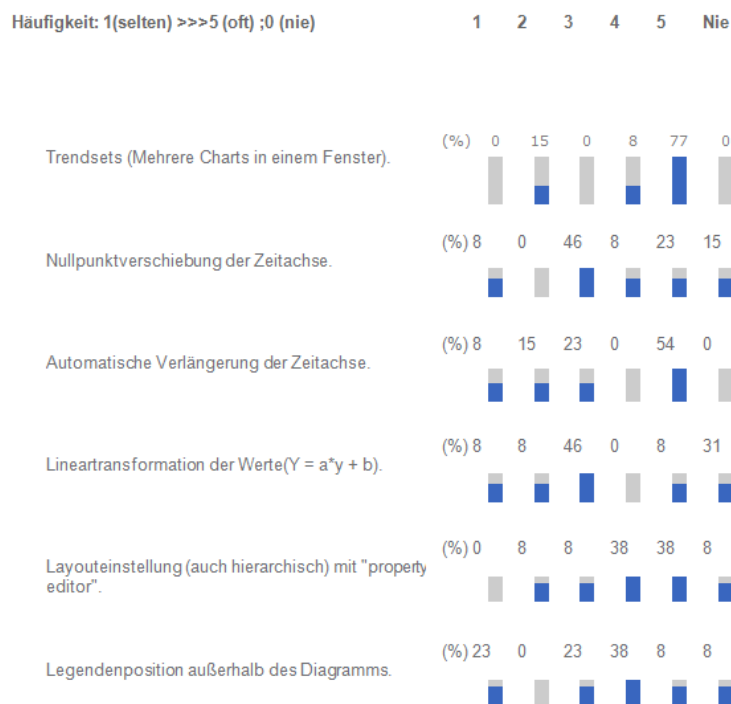


Abb. C.4 Passen Sie X/Y-Diagramme an?

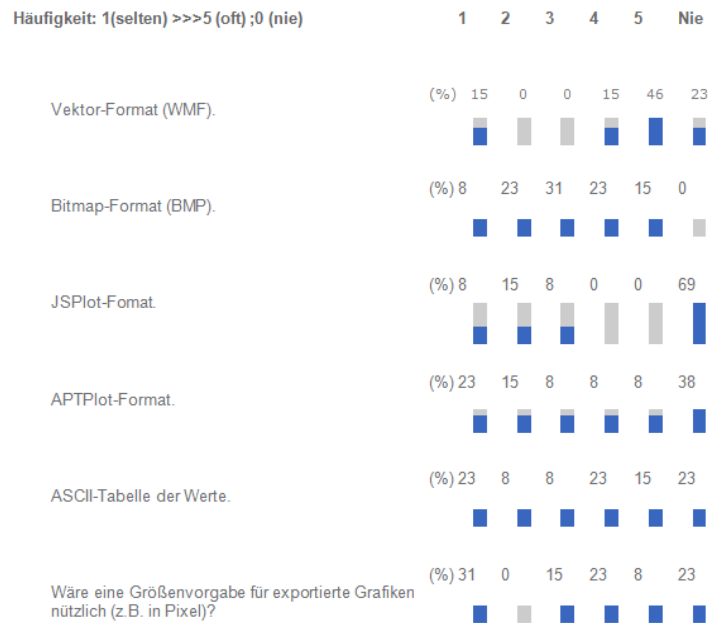


Abb. C.5 Nutzen Sie Datenexport für X/Y-Diagramme?

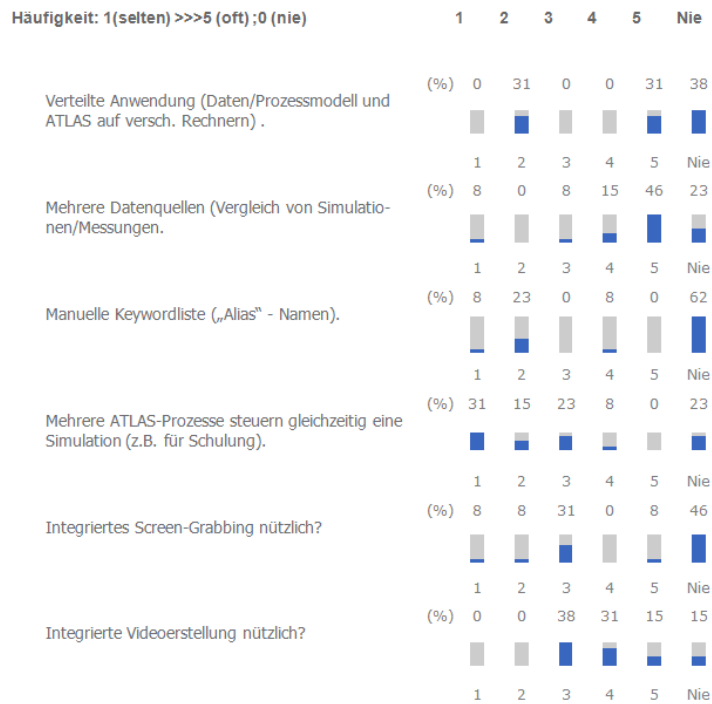


Abb. C.6 Nutzen Sie spezielle Funktionen?

**Gesellschaft für Anlagen-
und Reaktorsicherheit
(GRS) gGmbH**

Schwertnergasse 1
50667 Köln
Telefon +49 221 2068-0
Telefax +49 221 2068-888

Boltzmannstraße 14
85748 Garching b. München
Telefon +49 89 32004-0
Telefax +49 89 32004-300

Kurfürstendamm 200
10719 Berlin
Telefon +49 30 88589-0
Telefax +49 30 88589-111

Theodor-Heuss-Straße 4
38122 Braunschweig
Telefon +49 531 8012-0
Telefax +49 531 8012-200

www.grs.de