



D3.4.3: Implementierung von Langzeitarchivierungsdiensten

Teil 1: JANEME

(J-NetCDF Metadata Extractor)

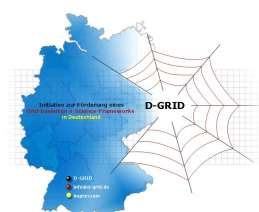
Version – 13.04.2011

Arbeitspaket 3

Verantwortlicher Partner - AWI

WissGrid

Grid für die Wissenschaft



Bundesministerium
für Bildung
und Forschung

WissGrid- Langzeitarchivierungsdienste, Teil 1: JANEME

Projekt: **WissGrid**

Teil des D-Grid Verbundes und der deutschen e-Science Initiative

BMBF Förderkennzeichen: 01|G09005A-G (Verbundprojekt)

Laufzeit: Mai 2009 - April 2012

Dokumentstatus: stabile Zwischenversion

Verfügbarkeit: öffentlich

Autoren: José Mejía, Bernadette Fritzsch (AWI)

Revisionsverlauf

Datum	Autor	Kommentare
15.03.2011	J. Mejía, B.Fritzsch	Erster Entwurf
13.04.2011	J. Mejía, B.Fritzsch	Neue Strukturierung. Aktualisierung der Tabellen, Bilder und URLs

Inhaltsverzeichnis

1 Einleitung.....	4
2 Anforderungen.....	6
3 Anwendungsfälle.....	7
4 Systemanforderungen.....	8
5 Modulbeschreibung.....	10
5.1 Quellcode-Architektur	10
5.2 Validierungsfunktionalitäten.....	13
5.3 Einschränkungen.....	14
6 Installation und Testing.....	16
7 Wartung und Support.....	18

1 Einleitung

Die Beschreibung von Daten durch Metadaten ist Voraussetzung für eine weitere Nachnutzung von Daten und damit Grundlage der Langzeitarchivierung. Dabei können Metadaten so komplex und umfangreich sein, dass ihre Erstellung für den Nutzer durchaus ein ernstzunehmendes Problem darstellt. Gerade das im Bereich der georeferenzierte Daten gebräuchliche ISO 199115/19139 stellt ein für die Wissenschaftler oft nicht leicht überschaubares Rahmenwerk auf, so dass sie auf Hilfe angewiesen sind. Andererseits werden bei der Modellierung in der Klimaforschung oft zwei Formate verwendet: NetCDF¹ und GRIB. Beide sind selbstbeschreibende binäre Formate, die also bereits eine Teilmenge der erforderlichen Informationen für das Metadatenprofil enthalten. Die Idee war nun, diese schon einmal abgespeicherten Informationen auszulesen und in das Metadatenprofil in die richtigen Felder einzusortieren, damit der Nutzer zumindest teilweise durch diesen automatisierten Prozess entlastet wird.

Dazu wurde im Rahmen von WissGrid am Alfred-Wegener-Institut für Polar- und Meeresforschung **JANEME** oder der **J-NetCDF Metadata Extractor** entwickelt. Dabei handelt es sich um ein Modul für Jhove2², einem international mittlerweile verbreiteten Framework für die Identifikation von Dateiformaten, die Extraktion von Metadaten als auch die Validierung, das auch über eine recht aktive Community verfügt. Durch seine modulare Struktur ist es leicht erweiterbar, wie am Beispiel des NetCDF-Moduls hier exemplarisch gezeigt wurde.

Dieses Dokument dient als Einleitung in JANEME auf einer rein konzeptionellen Ebene und beschreibt die wesentlichen Eigenschaften der im Rahmen von WissGrid entwickelten Software. Die ausführliche Dokumentation von JANEME als Jhove2-Modul ist nach den in der dortigen Community vereinbarten Standards erstellt worden. Die jeweils aktuell gehaltenen Modulspezifikationen *specNetcdf.pdf* und *specGrib.pdf* sind Bestandteil des Installationspakets und entsprechen dem offiziellen Modul-Dokumentationsformat. In diesem Dokument wird daher für weitergehende Informationen auf diese beiden Spezifikationen verwiesen.

¹ siehe <http://www.unidata.ucar.edu/software/netcdf/>

² <http://www.jhove2.org/>

2 Anforderungen

- JANEME soll auf Jhove2 basieren, um eine möglichst große internationale Community anzusprechen.
- JANEME soll die in NetCDF-Format der Versionen 3.0 und 4.0 gespeicherten header-Informationen auslesen und automatisch in verschiedene Metadatenprofile ablegen können.
- Es soll NetCDF-Format der Versionen 3.0 und 4.0 und GRIB unterstützen.
- Es soll auf weitere Formate leicht erweiterbar sein.
- JANEME soll flexibel den Einsatz neuer Metadatenprofile erlauben.

3 Anwendungsfälle

Bei den vorbereitenden Schritten für den Dateneingest von NetCDF-Files erstellt der Nutzer mit Hilfe von JANEME ein Metadatenfile. Dabei werden technische Metadaten (u.a. Validierung der NetCDF-Version) sowie inhaltliche Metadaten aus den Headern extrahiert und in ein vorhandenes Metadatenmodell eines LZA-Repositorys eingetragen. Je nach ausgewähltem Profil sind die im header abgespeicherten Informationen bereits vollständig³ oder dienen als Ausgangspunkt, um dann mit einem Metadateneditor die noch fehlenden Informationen manuell eintragen zu können.

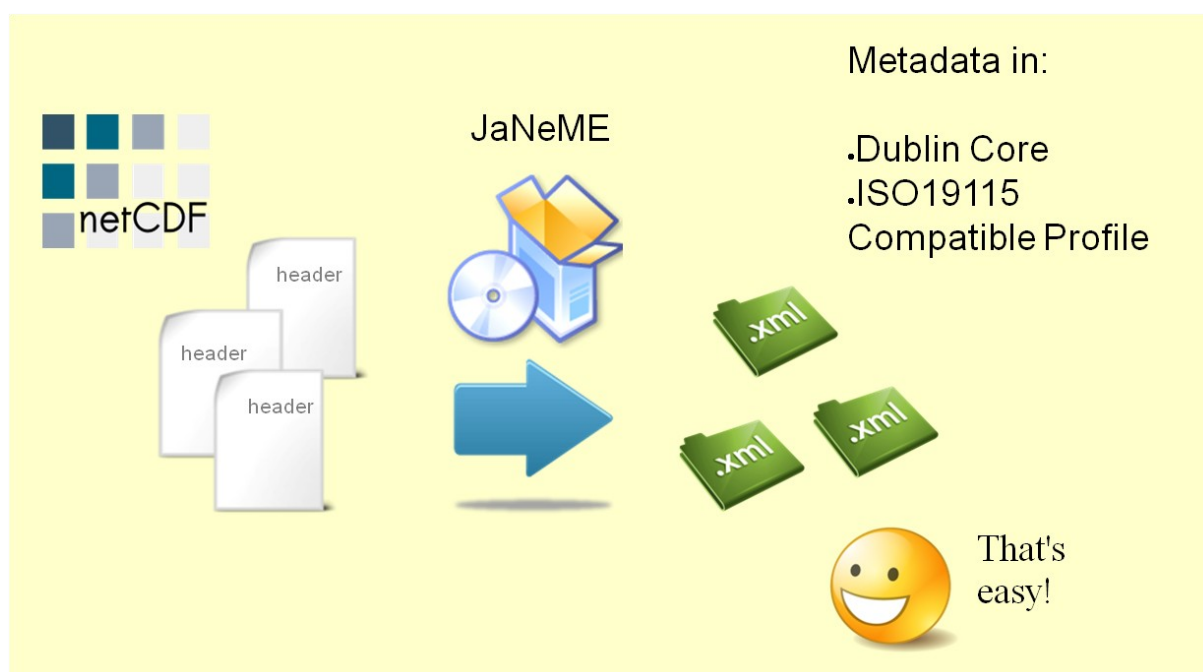


Abbildung 1: Use Case für Anwendung von JANEME

³ Wird Dublin Core verwendet, wird dies i.d.R., der Fall sein.

4 Systemanforderungen

JANEME ist plattformunabhängig und ist auf Windows Vista, Ubuntu 9.x und Solaris 5.10. getestet worden.

JANEME basiert so wie die ganze Jhove2-Applikation auf dem Spring Framework 2.5. Spring ermöglicht eine wesentliche Vereinfachung der notwendigen Java Entwicklungen und unterstützt die Übertragung der hier vorgestellten Arbeiten. Weitere Vorteile sind die Reduzierung der Größe des Quellcodes, Verbesserung der Testbarkeit, Vereinfachung in der Verwaltung der Abhängigkeiten und Verbesserung des Applikationsdesigns dank der in Spring verfügbaren Designpatterns.

Die Stärke der Verwendung des Spring Frameworks besteht in der sogenannten Inversion-of-Control, einem Mechanismus für die einfache Definition der Komponenten und ihrer Abhängigkeiten während ihres gesamten Lebenszyklus. So wird eine Komponente wie ein Formatmodul, Displayer oder eine Datenbank-JDBC-Verbindung in dem Framework registriert und durch ihre API zu einem späteren Zeitpunkt aufgerufen.

Die Anforderungen an das System werden durch die in Tabelle 1: Abhängigkeiten zusammengefassten Projektabhängigkeiten definiert.

Maven Artifact Id	Maven Group Id	Version	Maven Scope
netcdf-java-all	essi-unidata	4.1	compile
velocity	org.apache.velocity	1.6.2	compile
velocity-tools	org.apache.velocity	2.0-alpha1	compile
derby	org.apache.derby	10.5.3.0_1	compile
mail	javax.mail	1.4.1	compile
spring-jdbc	org.springframework	2.5.3	compile
spring-aop	org.springframework	2.5.3	compile
cglib	cglib	2.2	compile
asm	asm	3.2	compile
axis2-codegen	org.apache.axis2	1.5.1	provided
axis2-adb-codegen	org.apache.axis2	1.5.1	provided
axis2-adb	org.apache.axis2	1.5.1	provided
neethi	org.apache.neethi	2.04	provided

Tabelle 1: Abhängigkeiten

Da die notwendige Version der *netcdf-java* Bibliothek auf keinem maven Repository verfügbar ist, muss sie vorbereitend für die Installation manuell in das lokale maven Repository durch den folgenden maven Befehl importiert werden:

```
mvn install:install-file -Dfile=$JHOVE2_HOME/lib/netcdfAll-4.1.jar -DgroupId=essi-unidata -DartifactId=netcdf-java-all -Dversion=4.1 -Dpackaging=jar
```

5 Modulbeschreibung

5.1 Quellcode-Architektur

Der Quellcode und seine JUnit tests liegen in den Packages *src/main/java* bzw. *src/test/java*. Alle dazugehörigen Ressourcen findet man im Verzeichnis *src/main/resources/netcdf*.

Die nötigen Beans, ihr Geltungsbereich, die gegenseitigen Abhängigkeiten und ihre zugehörigen Ressourcen werden in *src/main/resources/netcdf/netcdf-config.xml* nach der Spring 2.5 Syntax deklariert.

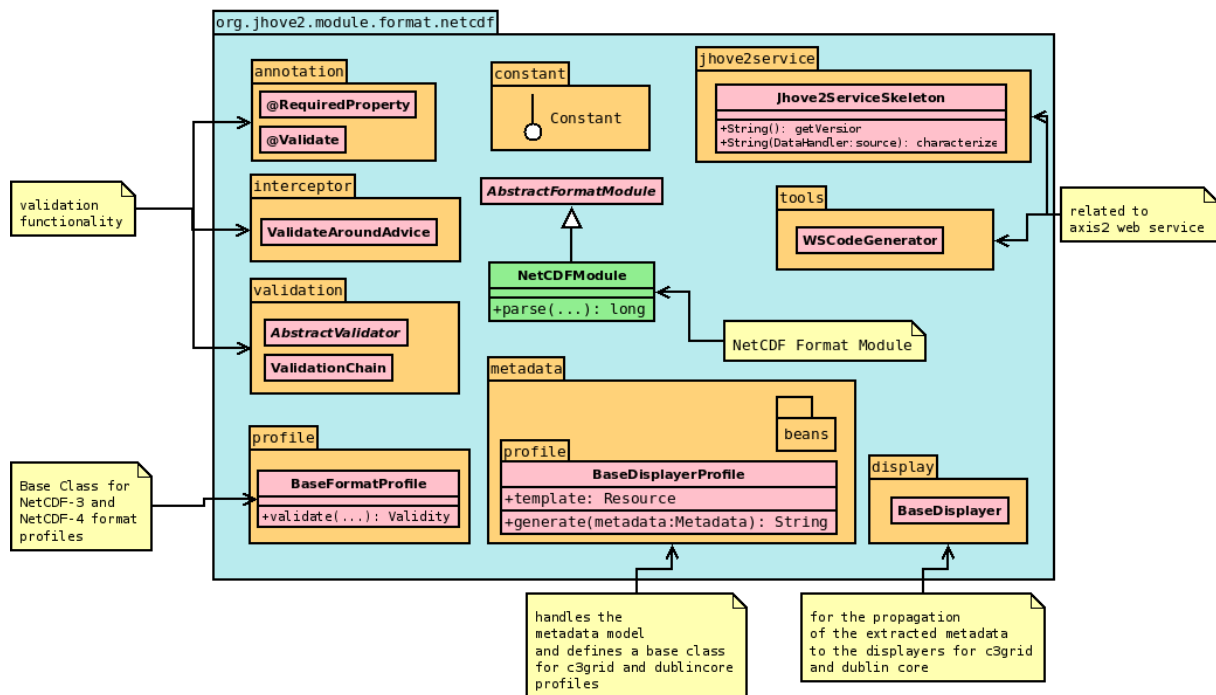


Abbildung 2: Klassendiagramm

In der Abbildung 2 ist die Organisation des Codes schematisch dargestellt. Im Package *org.jhove2.module.format.netcdf*, findet man die Klasse *NetCDFModule* – das Herz JANEMEs – und ihre anverwandten Subpackages.

Das Package *metadata* besteht aus den Subpackages *beans* und *profile*. Die von dem Fileheader extrahierten Attributwerte werden in einem *Metadata* Bean (aus dem Subpackage

beans) gespeichert, dessen Instanz in das *NetCDFModul* durch eine Spring-Bean-Referenz-Deklaration angelegt wird.

Die Implementierung enthält zwei Metadatenprofile, nämlich *C3Profile* und *DCProfile*. Die beiden sind Spring Bean Instanzen der Klasse *BaseDisplayerProfile* im Subpackage *metadata.profile* und für die Generierung der Metadatendarstellung durch das Ausfüllen eines *Apache Velocity* Templates mit den in *Metadata*-Beans gespeicherten Attributenwerten verantwortlich.

Die Profile können nur die Standardausgabe durch einen *AbstractDisplayer* erreichen. Für diesen Zweck liefert JANEME beispielhaft die Displayer *C3ProfileDisplayer* (identifiziert als *c3grid*) und *DCProfileDisplayer* (identifiziert als *dublincore*) als Spring Beans der Klasse *org.jhove2.module.format.netcdf.display.BaseDisplayer* mit. Der Endbenutzer kann dennoch die Displayer *JSON*, *Text* oder *XML* weiterverwenden.

Anpassungen an andere Metadatenprofile sind leicht möglich. Die Anwendung von JANEME ist damit flexibel, der Nutzer entscheidet zur Laufzeit, auf welches Metadatenprofil er die extrahierten Metadaten mappen will.

So kann mit den Aufrufen

```
./jhove2.sh -d c3grid -o output.xml /home/user/Beispiel.nc bzw.
```

```
./jhove2.sh -d dublincore -o output.xml /home/user/Beispiel.nc
```

die Metadateninformation des gleichen Files *Beispiel.nc* einmal im Hinblick auf das C3Metadatenprofil und einmal auf Dublin Core extrahiert werden.

Darüber hinaus werden die Templates, die Profile sowie die Displayer in der *netcdf-config.xml* Datei eingetragen.

Allgemeine Konstanten wie *keyword_delimiter*, *default_buffersize* und das enum *variable_recognized_attributes* werden im Interface *Constant* aus dem Package *constant* abgelegt.

Im Package *annotation* befindet sich der Quellcode für die Methoden-Annotationen *@RequiredProperty* und *@Validate*. Die erste wird bei Bean-Getters verwendet und liefert bei der Rückgabe eines Null-Wertes eine Warnung. Wird ein String-Returntyp erwartet, so wird *null* durch „*Please insert a value*“ in der Ausgabe ersetzt. Die zweite Annotation ist für Setters und zeigt an, dass der Setter-String-Parameter nach einer Validierungskette verifiziert sein soll.

Die Subpackages *interceptor* und *validation* bieten eine Verifikationsfunktionalität für Beans-Properties annotiert mit *@RequiredProperty* oder *@Validate*. Diese Eigenschaft wird im folgenden Abschnitt ausführlicher diskutiert.

Das Standard JANEME Mapping in *src/main/resources/netcdf.properties* definiert die Beziehungen zwischen den NetCDF globalen Attributen und den Bean-Properties durch Ausdrücke der Art

netcdf.metadata.{Attribut_name}= {Setter-Ausdruck relativ zu dem Metadata Bean}.

Als ein Beispiel würde die Definition

netcdf.metadata.acknowledgment=documentation.acknowledgment

bedeuten, dass das Attribut „*acknowledgment*“ mit dem Setter *Metadata.getDocumentation.setAcknowledgment()* zu speichern ist.

Dieses Mapping kann beliebig modifiziert werden, wobei dazu neu kompiliert werden muss. Als Alternative dazu bietet JANEME dem Endbenutzer die Möglichkeit, das Mapping durch eine Property-Datei zu steuern, deren Pfad in der System-Property *netcdf.mapping* angegeben werden muss (Beispiel: *JAVA_OPTS=-Dnetcdf.mappings=/home/user/myMapping*“).

In diesem individuell anpassbaren File kann dann auch das Schlüsselwörtertrennzeichen als regulärer Ausdruck mittels *netcdf.metadata.delimiters=[,\\n]+* festgelegt werden. Sollte die Property *netcdf.mappings* nicht definiert sein, wird das standard Mapping verwendet.

Am Ende der c3grid-Profil-Ausgabe werden all die Attribute und ihre Werte als Kommentar aufgelistet, für die kein Mapping definiert wurde. Dies ermöglicht Korrekturen der Ausgabe durch XSLT Transformationen oder weist darauf hin, dass Anpassungen in dem Mapping nötig sein könnten.

5.2 Validierungsfunktionalitäten

Mit Hilfe der AOP (Aspect Oriented Programming) Unterstützung ist eine Validierung in JANEME möglich. Die Klasse *ValidateAroundAdvice* aus dem Package *validation* verhält sich wie ein Interceptor für Getters und Setters in den Metadatenbeans, deren Instanzen durch Proxies gehandelt werden. Durch die Deklaration eines *BeannameAutoProxyCreator* 's wird dieses Verhältnis dem IoC Container mitgeteilt.

Die Methode *invoke* der Java-Klasse *ValidateAroundAdvice* kann vor und nach dem Aufruf der Bean-Target-Methode ablaufen. So kann überprüft werden, ob die begleitenden Annotationen *@RequiredProperty* und *@Validate* existieren, und ein entsprechender Validierungsprozess kann durchlaufen werden.

Die Annotation *@Validate* präzisiert den Namen einer Validierungskette, der mit dem *id* einer ValidationChain-Bean in der Spring-Konfiguration *netcdf-config.xml* übereinstimmt. Eine Validierungskette besteht aus einer ArrayList von Validatoren und einer *String*-Nachricht für den Fall, dass die Validation scheitert – wenn die Methode *isValid* der ValidationChain *false* zurückgibt. Dies ermöglicht das Testen der Konformität von Variablennamen mit einem bestimmten Vokabular, z.B. der CF Convention, in einer spezifischen Reihenfolge. Immer wenn die Methode *isValid* eines Validators *true* zurückgibt, sind keine weiteren Verifikationen mehr nötig. Bei einem Fehlschlag eines Validators wird eine weitere Verifikation durch den nachfolgenden Validator in der Kette erzwungen. So kann eine Kette *false* dann und nur dann zurückgeben, wenn alle ihre Validatoren *false* zurückgeben.

Ein Vorteil der Validierungskette besteht in der Wiederverwendung der Validatoren in verschiedenen Ketten.

Die Klassen für *ValidationChain*, Validatoren und ihre unterstützenden Klassen befinden sich im Subpackage *validation*.

Für die Verbreitung des Moduls ist wesentlich, dass die Erweiterung einer Validation in JANEME konzeptionell sehr trivial ist. Die weitere Entwicklung eines vorhandenen Vokabulars sowie die Integration eines neuen in das Modul sind möglich. Dieses Vokabular kann in einem enum, array oder sogar in einer Datenbank gespeichert werden. Exemplarisch ist das für den CFValidator, der eine portable Apache Derby Datenbank bestehend aus 1907 CF-Standardnamen und 145 Aliases anfragt.

5.3 Einschränkungen

Da es keine Standards für das Mapping von NetCDF auf ISO 19115 oder Dublin Core gibt, muss der Endbenutzer eventuell die entsprechenden Templates oder sogar den Quellcode anpassen.

6 Installation und Testing

Die Installation JANEMEs erfolgt nach dem in der Jhove2 offiziellen Anleitung⁴ erklärten Vorgehen und besteht aus den folgenden Schritten:

1. Java 6 Einrichten

- Überprüfung der aktuellen Java-Installation

Befehl: `java -version`

- Installation oder Aktualisierung Javas

Zum Download der Java-Software für Windows, Linux oder Solaris besuchen Sie <http://java.sun.com/javase/downloads/>. Diese Seite bietet Ihnen die Möglichkeit zum Download des Java SE Development Kit (JDK) oder der Java Runtime Environment (JRE). Die JRE reicht aus, um JHOVE2 benutzen zu können. Aber wenn Sie die Weiterentwicklung Jhove2s oder JANEMEs planen, laden Sie das JDK, die grundlegende Entwickler-Tools, herunter.

2. JANEMEs Archiv herunterladen und extrahieren

- JANEMEs ZIP-Archiv ist auf <http://aforge.awi.de/gf/project/jhove2/frs/> zum freien Download verfügbar.
- Verschiedene ZIP-Dienstprogramme sind auf den meisten Plattformen vorinstalliert. Das folgende Beispiel extrahiert eine JHOVE2 ZIP-Archiv in das aktuelle Verzeichnis: `unzip jhove2.zip`

3. Anpassung des JHOVE2 Skriptes, wenn nötig

4. Konfiguration von JHOVE2, wenn nötig

Mit dem Aufrufen

```
./jhove2.sh -d c3grid -o output.xml src/test/resources/examples/netcdf/classic.nc bzw.
```

```
./jhove2.sh -d dublincore -o output.xml src/test/resources/examples/netcdf/classic.nc
```

können Sie die Metadateninformation der Beispiel-Datei *classic.nc* einmal im Hinblick auf das C3Metadatenprofil und einmal auf Dublin Core extrahieren. Die standard Displayer *JSON*, *Text* und *XML* sind auch verfügbar.

⁴ Siehe http://bitbucket.org/jhove2/main/wiki/documents/JHOVE2-Users-Guide_20110222.pdf

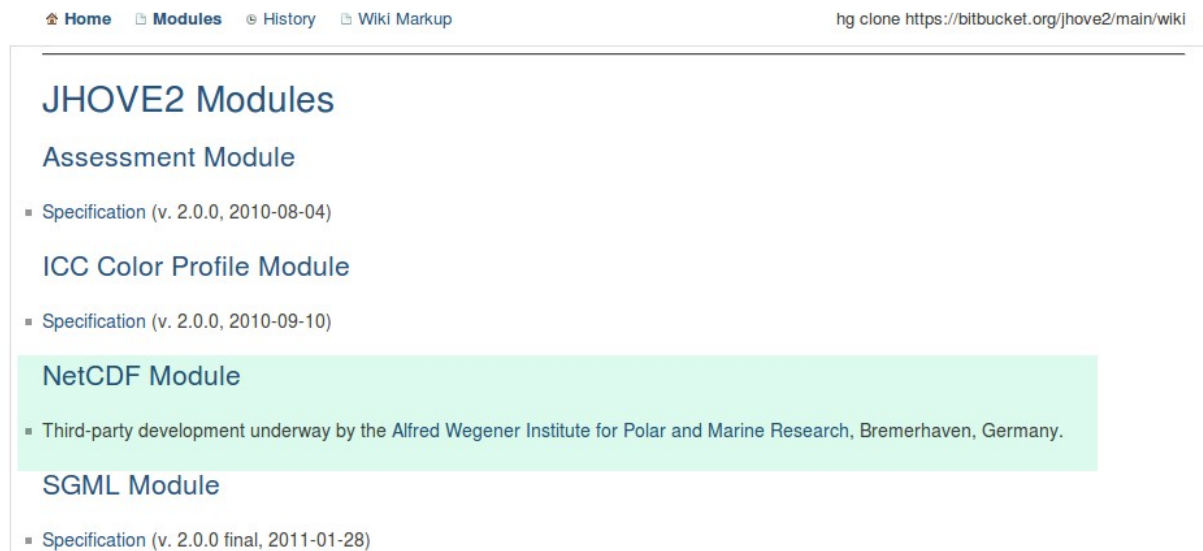
WissGrid- Langzeitarchivierungsdienste, Teil 1: JANEME

Das Installationspaket enthält Beispiele für NetCDF und GRIB Dateien unter *src/test/resources/examples/netcdf* und *src/test/resources/examples/grib* sowie die Modulspezifikationen *specNetcdf.pdf* und *specGrib.pdf*, die eine komplette Übersicht der Benutzung und technischen Details JANEMEs in der englischen Sprachen liefern.

7 Aktueller Stand

JANEME kann unter <http://aforge.awi.de/gf/project/jhove2/frs/> heruntergeladen werden.

Es ist ein unter der BSD-Lizenz freigegebenes Jhove2-Formatmodul und wird offiziell als Third-Party-Development bei Jhove2 erwähnt.



Home Modules History Wiki Markup hg clone <https://bitbucket.org/jhove2/main/wiki>

JHOVE2 Modules

Assessment Module

- Specification (v. 2.0.0, 2010-08-04)

ICC Color Profile Module

- Specification (v. 2.0.0, 2010-09-10)

NetCDF Module

- Third-party development underway by the Alfred Wegener Institute for Polar and Marine Research, Bremerhaven, Germany.

SGML Module

- Specification (v. 2.0.0 final, 2011-01-28)

Abbildung 3: Verweis auf JANEME auf der Jhove2-Webseite <https://bitbucket.org/jhove2/main/wiki/Modules>

8 Wartung und Support

JANEMEs Code wird von José Mejía Villar, M.Sc., vom Alfred-Wegener-Institut für Polar- und Meeresforschung in Bremerhaven, Deutschland gepflegt.

Der Entwickler kann für Supportanfragen unter <mailto:Jose.Mejia@awi.de> erreicht werden. Hier erhalten Sie fachliche Unterstützung in allen Fragen rund um JANEME und zur Fehlerbehebung und können uns Feedback auf Englisch, Spanisch oder Deutsch geben.