



# Durchführung der Tests für AP 2.5

**Anne Milbert (AEI)**

**Gevorg Poghosyan (FZK/KIT)**

**Tibor Kálmán (GWDG)**

**Tobias Lindinger (LMU)**

**Mathilde Romberg (FZJ)**

**Rebecca Breu (FZJ)**

16. Juli 2009

## INHALTSVERZEICHNIS

<b>1</b>	<b>EINLEITUNG .....</b>	<b>3</b>
<b>2</b>	<b>MIDDLEWARE-ÜBERGREIFENDE TESTS .....</b>	<b>4</b>
2.1	TESTSZENARIEN FÜR ADAPTER .....	4
2.2	TESTSZENARIEN FÜR DIE DATENBANK .....	6
<b>3</b>	<b>VO-BASIERTE- UND INTEGRATIONSTESTS.....</b>	<b>9</b>
3.1	TESTSZENARIEN FÜR DIE DATENBANK .....	9
3.2	TESTSZENARIEN FÜR DAS FRONTEND .....	10
<b>4</b>	<b>PERFORMANZ-TESTS .....</b>	<b>19</b>
4.1	LASTTESTS .....	19
4.2	VERTEILUNGSTESTS .....	22
<b>5</b>	<b>ZUSAMMENFASSUNG .....</b>	<b>23</b>
<b>6</b>	<b>REFERENZEN .....</b>	<b>24</b>

## **1 Einleitung**

Nach der erfolgten Implementierung der D-MON Komponenten wird die Software nun anhand der vorher definierten Tests [2] ausführlich getestet. Die Ergebnisse helfen dabei etwaige bestehende Engpässe oder zu verbessernde Teile zu identifizieren. In der Regel wurden notwendige Anpassungen direkt vorgenommen und die Tests wiederholt. Die jeweils mit der aktuellsten Software erzielten Ergebnisse werden hier vorgestellt.

Dieses Dokument beschreibt die Durchführung der in AP2.4 [2] spezifizierten Tests und listet die erzielten Ergebnisse auf. Daher ist dieses Dokument analog zu AP2.4 aufgebaut: Jedes Kapitel enthält die Ergebnisse des korrespondierenden Kapitels aus dem Definitionsdokument. Zunächst werden in Kapitel 2 die Middleware-übergreifenden Tests der Adapter und der Datenbank ausgeführt. Kapitel 3 konzentriert sich auf die VO-basierten Aspekte in Bezug auf die Datenbank und die Nutzerschnittstellen. Testergebnisse zur Performanz des Systems werden in Kapitel 4 behandelt. Die erzielten Ergebnisse werden zum Abschluss in Kapitel 5 zusammengefasst.

## 2 Middleware-übergreifende Tests

### 2.1 Testscenarien für Adapter

Die Adapter-Ebene lässt sich in die Module der einzelnen Middlewares zergliedern. Auf Ebene der Middlewares gliedern sich diese dann weiter in die XSL Stylesheets für die SQL Einfügeoperationen und in über `<xsl:include>` geladene Templates bzw. Stylesheets. Diese sollen in diesem Kontext als Units, also als kleinstmögliches Element dessen Funktionalität getestet werden soll, verstanden werden.

#### 2.1.1 Modultests

Die Modultests sind analog zu den drei unterstützten Middlewares aufgegliedert. Auf dieser Ebene werden für jedes Modul die Testscenarien für die Unit Tests durchgeführt. Tests gelten als erfolgreich, wenn sie die erwarteten Testergebnisse erzielt haben.

Im Folgenden werden die Testscenarien gelistet und dann die Durchführung und Ergebnisse pro Middleware aufgeführt..

#### Testscenarien

- 1) Abfrage an lokale Site-Information Server bzw. zentrale D-Grid Informationsserver. Für jede Middleware wird ein entsprechendes Abfragesystem eingesetzt:

##### **Globus MDS4 – Durchführung:**

*Die Abfrage des MDS erfolgt über das Globus-Tool "wsrf-query" unter Angabe der URI des Site spezifische MDS Servern.*

##### **Globus MDS4 – Ergebnisse:**

*Wsrf-Query arbeitet sowohl mit den getesteten Site lokalen als auch Top-Level MDS Instanzen fehlerfrei.*

##### **glite BDII – Durchführung:**

*Durch das `ldapxml.pl` Scriptprogramm unter Angabe von Fully Qualified Domain Name (FQDN) des BDII Site-Information Servern, werden Daten abgefragt und ins XML Format umgewandelt. Hier wird für alle Verbindungen „o=grid“ als Stammpunkt: Basis-DistinguishedName(DN) und Port 2170 benutzt.*

##### **glite BDII – Ergebnisse:**

*Abfrage von Daten aus top als auch einzelne site BDII Servern, Umwandlung von Lightweight Directory Access Protocol (LDAP) ins XML, laufen ohne Fehler. Ergebniss wird in Directory Services Markup Language (DSML) Schema ins Standart output gezeigt oder ins Datei gespeichert.*

##### **Unicore CIS – Durchführung:**

*Die Abfragen an den UNICORE CIS erfolgen über Kommandos des UNICORE Command Line Client UCC. Unter Angabe der URI des CIS zentralen Servern werden über `cis-showallinfo` alle verfügbaren Informationen der in diesem CIS registrierten Sites ausgelesen.*

**Unicore CIS – Ergebnisse:**

*Die Ergebnisdaten stehen in einem GLUE2.0 XML Dokument zur Verfügung. Fehlermeldungen sind nicht aufgetreten.*

- 2) Generierung von XML aus dem Output des Informationsdienstes einer Middleware mit Hilfe der implementierten Adapter, falls dieser Schritt notwendig ist.

Manuelle Überprüfung des Ergebnisses mit Hilfe von XML Validatoren und Validierung gegen existierende Schemata. Dieser Teilttest muss mit unterschiedlichen Datenquellen durchgeführt werden, um möglichst viele potentielle Fehler finden zu können.

**Globus MDS4 – Durchführung:**

*Eine Umwandlung der Daten in das XML Format ist nicht notwendig, da wsrf-query bereits einen XML-Baum zurückliefert. Sicherheitshalber wird der erhaltene Baum mit Hilfe von XML Validatoren überprüft.*

**Globus MDS4 – Ergebnisse:**

*Die Überprüfung des Baumes auf syntaktische Korrektheit, lieferte einige kleinere Fehler, die jedoch die Funktionalität des XSLT-Prozessors nicht beeinträchtigen. Eine Korrektur dieser Fehler kann wahrscheinlich nur durch die Globus Entwickler vorgenommen werden, so dass zumindest kurzfristig mit den Fehlern gelebt werden muss.*

**glite BDII – Durchführung:**

*Umwandlung von LDAP ins XML DSML, laufen unmittelbar nach dem Server Abfrage und wird durch gleichem Idapxml.pl Scriptprogram erledigt.*

**glite BDII – Ergebnisse:**

*In LDAP vorhandene Baum abhängige Hierarchien oder Einschränkungen, z.B. VO spezifische Informationen wie Zugriffsrechte, sind syntaktisch nicht ins DSML übertragbar. XML Ergebniss enthält vereinfachte Baumstruktur, aber durch Inhalt von DN Wiederherstellung von Hierarchien ist möglich.*

**Unicore CIS – Durchführung:**

*Eine Umwandlung der Daten in das XML-Format ist nicht notwendig, da der CIS die Daten in gültigem XML-Format ausliefert..*

**Unicore CIS – Ergebnisse:**

...

- 3) XML2SQL: Generieren von SQL Statements zum Einfügen / Update der extrahierten XML Datensätze in die Datenbank. Die Syntax des Outputs gilt als korrekt, wenn die Datenbank keine Fehler zurückliefert.

Die Zuordnung von XML Attributen und Tags zu Datenbank-Feldern muss manuell überprüft werden, insbesondere sollte sie konform zu GLUE 2.0 und für alle Adapter homogen sein. Aus diesem Grund wird der Semantik Test der einzelnen Adapter in den Konsistenzcheck der

Datenbank verlagert, da ansonsten keine middleware-übergreifende, einheitliche Darstellung der Daten in der Datenbank gewährleistet werden kann.

**Globus MDS4 – Durchführung:**

*Es wurden mehrere Datenquellen, darunter auch der Toplevel MDS des D-Grid mit Hilfe des Adapters abgefragt und SQL-Insert Kommandos erzeugt.*

**Globus MDS4 – Ergebnisse:**

*Es traten keine Fehler auf.*

**glite BDII – Durchführung:**

*Daten aus Zentralen BDII des D-Grid umwandlung ins SQL Queryies durch bdii.xlst, werden in Mehrere Test Datenbanken Inseriert*

**glite BDII – Ergebnisse:**

*Es traten Fehler beim Inserieren, für die Datenbank, wenn unvollständigem Schema oder extra FremdkKeys in Datenbank vorhanden waren. Fehlende Felder wegen Implementierung von unvollständigen Schema sind durch auskommentieren in in bdii.xslt beseitigt. Fehler wegen FremdKey werden lokalisiert mit Umwandlung einzelne Site-BDII Dateien, die jedoch sind abhängig von Fehlerhafte Konfiguration einzelne Sites von lokalen Administratoren.*

*Die Fehler beeinträchtigen nicht Funktionalität des XML ins SQL Prozesses und für die Sites mit Korrekt Datensätze in BDII treten bei keiner Fehler und Daten stehen in Datenbank in vorgesehenen Felder zur verfügung.*

**Unicore CIS – Durchführung:**

*Es wurde der D-MON-CIS, welcher die Informationen der CIPs mehrerer Sites liefert, mithilfe des Adapters abgefragt. Aus den so erhaltenen erzeugte der Adapter SQL-Insert Kommandos.*

**Unicore CIS – Ergebnisse:**

*Es traten keine Fehler auf.*

## **2.2 Testszzenarien für die Datenbank**

Das GLUE 2.0 Datenbankschema ist so konzipiert, dass bereits beim Einfügen von Datensätzen Konsistenzprüfungen durchgeführt werden. So werden beispielsweise Datensätze, die keinen Bezug auf andere, übergeordnete Datensätze haben, bereits durch die Fremdschlüsselprüfung zum Zeitpunkt des Einfügeversuchs durch die Datenbank abgelehnt. Des Weiteren werden feldspezifisch einige Prüfungen auf bestimmte Formate und Typen durchgeführt. Geprüft wird daher nur, ob die entsprechenden Daten aus den Middleware spezifischen Monitoring Systemen konsistent in das GLUE 2.0 Format konvertiert werden und ob die durch den Trigger generierten VO Informationen konsistent und richtig sind.

### 2.2.1 Modultests

Die D-MON-Datenbank ist das zentrale Modul zur übergreifenden Sammlung und Bereitstellung der Monitoringdaten an andere Grid-Komponenten.

#### TestszENARIO 1: Anlegen der Datenbank, Erzeugen der Tabellen

- 4) Das Initialisieren der Datenbank übernehmen Skripte, deren Erfolg daran gemessen werden kann, ob im Anschluss entsprechende Anfragen an die lokale Datenbank gestellt werden können.

Des Weiteren müssen die nötigen Nutzer Accounts in der DB angelegt und mit entsprechenden Berechtigungen versehen sein. Auch dies kann mit entsprechenden Testabfragen / Testeinträgen überprüft werden.

##### *Durchführung*

*Anlegen der Datenbank auf einem nicht vorkonfiguriertem System mittels des im Paket enthaltenen Setup- Skriptes. Lediglich die Installation der MySQL Datenbank Pakete in der Version 5.1 oder höher über den Paket-Manager der Distribution wurde zuvor manuell durchgeführt. Dieser Punkt ist in der Installationsanleitung des D-MON Datenbank Paketes vermerkt.*

##### *Ergebnisse*

*Das Installations-Skript legt wie erwartet alle Nutzer, Tabellen, Views, Events und Berechtigungen korrekt an; scheiterte aber beim Anlegen der VO-Management Tabellen, welche die MySQL Federated Engine verwenden. Der Grund für diesen Fehler war die in SLES 10 standardmäßig deaktivierte Federated Engine, während auf dem Entwicklungssystem, das auf Debian Lenny basiert, diese Engine aktiviert war. Ein Eintrag in die Konfigurations-Datei der Datenbank konnte den Fehler beheben. Die Dokumentation wurde entsprechend ergänzt. Ein weiterer Fehler wurde bei der Kontrolle der angelegten Tabellen sichtbar: Das Encoding der Datenbank war von SuSE auf Schwedisch gestellt, so dass Umlaute nicht korrekt dargestellt werden konnten. Ein entsprechender Eintrag in der Konfigurationsdatei der Datenbank löste auch dieses Problem. Der Eintrag wurde ebenfalls in die Dokumentation aufgenommen.*

#### TestszENARIO 2: Konsistenzcheck

- 5) Die Daten der Informationsdienste müssen einheitlich und korrekt in das GLUE 2.0 Schema konvertiert und in die Datenbank eingefügt werden. Um die korrekte Funktionsweise der beteiligten Komponenten zu verifizieren müssen Testadapter angebunden werden und die von ihnen eingefügten Daten in der Datenbank manuell kontrolliert werden. Als Referenzkriterien können hierzu die in GLUE 2.0 gegebenen Feldbeschreibungen und Beispiele dienen.

##### *Durchführung*

*Eine automatisierte Überprüfung aller Einträge auf syntaktische und semantische Korrektheit ist nicht möglich. Alle Datensätze, die in Testläufen der Adapter aus den drei verschiedenen Middleware-Instanzen gewonnen wurden, müssen manuell auf ihr korrektes Mapping in das Glue 2.0 Schema überprüft werden.*

##### *Ergebnisse*

*Aufgrund der unterschiedlichen zugrundeliegenden Schemata der Middleware Informationssysteme, ist es zwar möglich semantisch gleichbedeutende Einträge einem*

*entsprechenden Glue 2.0 Feld zuzuordnen, eine automatisierte Syntaxanpassung wäre aber falls sie überhaupt möglich ist, sehr aufwändig. Daher existieren Attribute in der Datenbank, deren Inhalt abhängig von der entsprechenden Quelle des Datensatzes syntaktisch unterschiedlich dargestellt werden. Ein Beispiel hierfür sind unter anderem Bezeichner für Sites, zum Beispiel LRZ-München im BDII und LRZ im MDS.*

### 3 VO-basierte- und Integrationstests

#### 3.1 Testscenarien für die Datenbank

Es werden feldspezifisch Prüfungen auf bestimmte Formate und Typen durchgeführt. Geprüft werden muss insbesondere auch, ob die durch den Trigger generierten VO Informationen konsistent und richtig sind.

##### 3.1.1 Modultests

##### Testscenario 1: Anbindung der VO-Management DBs

- 6) Das Initialisieren der Datenbank übernehmen Skripte, deren Erfolg daran gemessen werden kann, ob im Anschluss an ihre Ausführung entsprechende Anfragen an die angebotenen VO-Management Datenbanken gestellt werden können.

###### *Durchführung*

*Das Installations-Skript der Datenbank beinhaltet das Erzeugen eines Events, welches periodisch Informationen aus dem VO-Management in die lokale Datenbank schreibt. Enthält die lokale Datenbank anschließend die identischen Daten, wie das entfernte VO-Management System gilt der Test als erfolgreich.*

###### *Ergebnisse*

*Die Daten in lokaler und entfernter Datenbank waren nach einmaligem Durchlauf des Triggers in beiden Datenbanken identisch. Auch ein mehrfacher Aufruf des Skriptes und manuelle Änderungen an der lokalen Tabelle änderten am Erfolg des Triggers nichts.*

##### Testscenario 2: UPDATE

- 7) Der Bestand der D-MON Datenbank soll regelmäßig und automatisch aktualisiert werden. Für den Bereich der vertikalen Integration geschieht dies auf Basis von Datenbank Triggern, die beim Einfügen / Update von Datensätzen durch die Adapter aktiviert werden. Die Zuordnung von Services / Ressourcen zu VOs in der GLUE 2.0 Instanz muss nach dem Einfügen / Update von Datensätzen durch die Adapter manuell überprüft werden.

###### *Durchführung*

*Während diesem Test werden Datensätze, welche von den jeweiligen Adaptern aus den existierenden Monitoring Systemen extrahiert und transformiert wurden, in die Datenbank geschrieben. Anschließend sollte für jeden eingefügten ServiceEndpoint pro VO ein entsprechender Eintrag in der Tabelle GLUE20.AccessPolicy erzeugt worden sein, falls diese VO Zugriffsrechte auf den assoziierten Service hat. Dieser Eintrag wird von einem Trigger automatisch erstellt und aktuell gehalten. Die Inhalte der Tabelle müssen anschließend manuell mit dem VO Management und den eingefügten Daten abgeglichen werden.*

###### *Ergebnisse*

*Es wurden alle Einträge in die Tabelle durch den Trigger korrekt gesetzt, sofern die Benennung der Dienste im Middleware eigenen Monitoring-Dienst und im GRRS identisch waren. Unterschiedliche Benennung der Ressourcen und Dienste in den genannten Systemen führt zu fehlenden Einträgen in der GLUE20.AccessPolicy Tabelle und damit zu*

*unvollständigen VO-Views. Um vollständige VO-Views zu erzeugen müssen die Admins der Sites angehalten werden, die korrekten Bezeichner für ihre Ressourcen, so wie sie im GRRS gemeldet sind, in ihren Monitoring-Systemen einzutragen.*

- 8) Einträge in der Tabelle GLUE20.AccessPolicy, deren korrespondierender Eintrag im GRRS gelöscht wurde, können aus technischen Gründen nicht per Trigger in der D-MON Datenbank entfernt werden. Stattdessen existiert ein SQL Skript, das periodisch ausgeführt werden muss und veraltete Daten aus der Datenbank löscht. Dieses Skript wird dadurch getestet, dass Einträge der Adapter nicht aktualisiert werden und Berechtigungen im GRRS verändert werden. Das Ergebnis muss manuell überprüft werden.

*Durchführung*

*Zum Test dieses Bereinigungs-Mechanismus wurden einige Datensätze nicht mehr aktualisiert.*

*Ergebnisse*

*Der MySQL Scheduler aktivierte das Bereinigungs-Skript planmäßig. Alle Datensätze die ihre TTL überschritten hatten, wurden wie erwartet gelöscht.*

### **TestszENARIO 3: Generieren von VO-spezifischen Sichten**

- 9) Die erstellten DB-Views müssen auf ihre Richtigkeit überprüft werden. Dafür müssen für einige beliebige VOs die in deren View enthaltenen Datensätze manuell auf Korrektheit überprüft werden.

*Durchführung*

*Zunächst wird eine beliebige VO-View ausgewählt. Anschließend werden alle Ressourcen, Services und mit ihnen verknüpften Entitäten mit Hilfe eines GRRS-Auszuges überprüft. Dieser Test wurde mit mehreren VO-basierten Sichten durchgeführt.*

*Ergebnisse*

*Alle überprüften VO-Views wurden korrekt generiert. Während der Tests fiel jedoch auf, dass im VOMS VO Einträge für die VO Viola fehlten, aber im GRRS vorhanden waren und auch Berechtigung besaßen, Ressourcen zu verwenden. Darauf hin wurde die Komponente zum Import bestehender VOs so abgeändert, dass existierende VOs aus beiden Systemen übernommen werden. Das Problem wurde weitergegeben, die Inkonsistenz der VO-Managementsysteme aber bisher nicht behoben.*

## **3.2 TestszENARIEN für das Frontend**

### **3.2.1 Funktionalität**

Im Rahmen der Web-Applikation wurden die Tests programmatisch umgesetzt. Dies ermöglicht die Durchführung aller Tests während der Installation der Applikation. Wenn einer der Tests fehlschlägt, schlägt auch die Installation der Anwendung fehl. So kann vermieden werden, dass im Produktionsbetrieb Fehler auftreten, die durch fehlerhafte Installation oder durch Missachtung der in der Dokumentation definierten Voraussetzungen für die Lauffähigkeit der Web-Applikation (z.B. Installation von Softwarepaketen, auf die die Anwendung zugreift) auftreten koennen.

Durch den Einsatz von drei verschiedenen Frameworks (Portlet API, MyFaces/IceFaces und dem Spring Framework war es nicht immer möglich sich von allen Einflussgrößen unabhängig zu machen, wie es normalerweise in Softwaretests üblich wäre.

Die entwickelten Testsznarien simulieren sowohl den Zustand nach dem initialen Laden, als auch die korrekte Verarbeitung von User Events, also verschiedenen Aktionen, die von einem Nutzer/Besucher der Portalseite durchgeführt werden können. Inhaltlich beziehen sich die Tests im Bereich der Funktionalität hauptsächlich auf die Validität der dargestellten Informationen/Daten.. Es soll jedes einzelne Portlet in speziell auf seine Funktionalität zugeschnittenen Testsznarien überprüft werden.

Da die hier beschriebenen Szenarien nur dann aussagekräftig sind, wenn gewährleistet werden kann, dass die kleinstmögliche Einheit (in diesem Falle Methoden) des Programms reibungslos und korrekt arbeiten. Aus diesem Grunde wurden vor Durchführung der beschriebenen Tests noch diverse Unit-Tests<sup>1</sup> durchgeführt, deren Ergebnisse im Anhang zu finden sind. Um die Unit-Tests von der Datenbank unabhängig zu machen wurde diese durch ein Mock-Objekt ersetzt. Da ein Mock-Objekt keine Echt Daten zurückliefert sondern vorher zum Testfall passend festgelegte Werte, kann so genau ueberprüft werden, ob die die Daten korrekt abgefragt und transformiert werden. Im Rahmen der Modultests wird dann wieder mit der Datenbank gearbeitet. Im Rahmen der Modultests wird dann wieder mit der echten Instanz der Datenbank gearbeitet.

Ferner wurde im Rahmen der Integrationstests<sup>2</sup> der mit der OGSA-DAI Software ausgelieferte Client von Tobias Lindinger um den VO-Parameter erweitert und jeweils zur Überprüfung der Konsistenz der Daten den im Frontend ausgegebenen Daten gegenübergestellt. Ferner wurden alle möglichen durch User Events ausgelösten Funktionalitäten der verschiedenen Portlets automatisiert mit Hilfe des Selenium IDE Servers überprüft.

#### **TableViewPortlet:**

Dieses Portlet enthält eine Dropdownliste aller in der D-MON Datenbank enthaltenen VO-Views und eine Tabelle, die diese VO-Views anzeigt. Initial wird die VO-View der Tabelle ComputingService angezeigt.

Die Applikation arbeitet komplett dynamisch, was bedeutet, dass alle in der Tabelle dargestellten Daten „on the fly“ geladen werden. Die Applikation ermittelt über eine erste Abfrage alle Tabellen, die für diese VO und diese Rolle angezeigt werden dürfen und baut bei Auswahl eines Eintrags aus der Dropdownliste dynamisch aus den mitgelieferten Informationen die tabellarische Sicht auf die Daten auf. Das ermöglicht das Erweitern der Datenbank um neue Tabellen bzw. die in der Datenbank enthaltenen Tabellen um weitere Felder sowie das Erweitern des Rollenkonzepts ohne die Programmlogik der Applikation anpassen zu müssen.

---

<sup>1</sup> Der **Modultest** (auch **Komponententest** oder [engl. unit test](#)) ist Teil des [Softwaretests](#). Er dient zur [Verifikation](#) der [Korrektheit](#) von Modulen einer [Software](#), z.B. von einzelnen [Klassen](#). Nach jeder Änderung sollte durch Ablauf aller [Testfälle](#) nach [Programmfehlern](#) gesucht werden. (Wikipedia, <http://de.wikipedia.org/wiki/Unit-Test>)

<sup>2</sup> Der Begriff **Integrationstest** bezeichnet in der [Softwareentwicklung](#) eine aufeinander abgestimmte Reihe von Einzeltests, die dazu dienen, verschiedene voneinander abhängige Komponenten eines komplexen [Systems](#) im Zusammenspiel miteinander zu testen. Die erstmals im gemeinsamen Kontext zu testenden Komponenten haben jeweils einen [Unit-Test](#) erfolgreich bestanden und sind für sich isoliert fehlerfrei funktionsfähig.(Wikipedia, <http://de.wikipedia.org/wiki/Integrationstest>)

Table view Portlet

Tabelle auswählen:

ID	name	description	distributed	adminDomainID	informationProvider	sourceAddr	insertTime
LRZ	Leibniz-Rechenzentrum Garching	Leibniz-Rechenzentrum (LRZ)	0	LRZ	MDS4	120.187.254.30	2009-02-23 14:00:04.0

1 Eintraege gefunden, 1 Eintraege werden angezeigt, von 1 bis 1. Seite 1 / 1

Navigation: [Home] [Previous] [Next] [End]

**Abb. 1: Table view Portlet**

**Testszenerien:**

10) Korrekte Darstellung der Daten und der VO-Views beim initialen Laden der Seite:

- Korrekte Anzeige aller möglichen VO-Views in der Dropdown Liste
- Korrekte Anzeige der Daten der VO-View der Tabelle ComputingService

Durchführung: Als Unit- und Modultest mit JUnit und Spring Unit Tests

Korrekte Anzeige aller möglichen VO-Views in der Dropdown Liste: Testen des Absendens von Abfragen an den Client und der Transformation der Daten vom ResultSet zu einer Liste von SelectItems.

Unit-Test:

Testklasse: test.unit.de.dgrid.dmon.portal.dao.TestTableDao

Getestete Klasse(n): de.dgrid.dmon.portal.dao.TableDao  
de.dgrid.dmon.portal.dao.impl.TableDaoImpl

Modultest:

Testklasse: test.module.de.dgrid.dmon.portal.dao.TestTableDao

Getestete Klasse(n): de.dgrid.dmon.portal.dao.TableDao  
de.dgrid.dmon.portal.dao.impl.TableDaoImpl

Korrekte Anzeige der Daten der VO-View der Tabelle ComputingService: Test auf die fehlerfreie Transformation der initial geladenen Tabelleninhalte.

Unit-Test:

Testklasse: test.unit.de.dgrid.dmon.portal.dao.TestTableViewDao  
test.unit.de.dgrid.dmon.portal.util.TestColumnDataModellImpl

Getestete Klasse(n): de.dgrid.dmon.portal.dao.TableViewDao  
de.dgrid.dmon.portal.dao.impl.TableViewDaoImpl

Modultest:

Testklasse: test.module.de.dgrid.dmon.portal.dao.TestTableViewDao  
test.module.de.dgrid.dmon.portal.util.ColumnDataModellImpl

Getestete Klasse(n): de.dgrid.dmon.portal.dao.TableViewDao  
de.dgrid.dmon.portal.dao.impl.TableViewDaoImpl

Ergebnisse:

Korrekte Anzeige aller möglichen VO-Views in der Dropdown Liste:

Unit- und Modultests wurden erfolgreich durchgeführt (siehe Abb. 2 und Anhang).

Korrekte Anzeige der Daten der VO-View der Tabelle ComputingService:

Unit- und Modultests wurden erfolgreich durchgeführt. Details zu den Testergebnissen können dem Testbericht im Anhang entnommen werden.

- 11) Korrekt gesetzte VO Darstellung der Daten und einer VO-View bei Auswahl einer neuen VO aus der Dropdownliste:

- Neuladen der Tabelle und korrekte Anzeige der Daten aus der D-MON Datenbank

Durchführung: Als Integrationstest mit JWebUnit/Selenium IDE

Testklasse:

`test.integration.web.portlets.SeleniumTestTableChangeEvent`

Ergebnisse:

Das korrekte Nachladen der Daten bei Auswahl einer neuen VO-View aus der Dropdownliste konnte erfolgreich getestet werden.

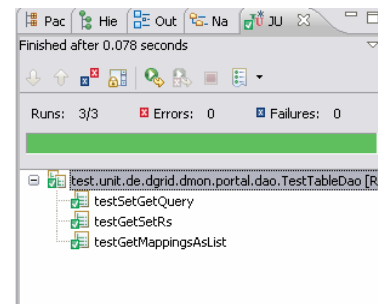


Figure 1 Ergebnisse TestTableDao

- 12) Per Mausklick auf Primär- und Fremdschlüsselfeldern der Tabelle muss ein Event ausgelöst werden und der Feldname und –inhalt des selektierten Feldes müssen mit dem Event übertragen werden:

- Mausklick auf ein Primär- oder Fremdschlüsselfeld muss ein Event auslösen
- Feldname und –inhalt des geklickten Feldes müssen korrekt übertragen werden

Durchführung: Als Integrationstest mit JWebUnit/Selenium IDE

Testklasse: `test.integration.web.communication.SeleniumTestInterPortletCommunication`

Ergebnisse:

Funktionalität konnte erfolgreich getestet werden. Mausklick auf verlinktes Feld führt zum Nachladen der Daten für das GMap Portlet und das DetailedView Portlet.

### DetailedViewPortlet:

Das Portlet ist eine Java Applikation, die alle VO-Views anzeigt, die in irgendeiner Form inhaltlich mit dem Feldinhalt verknüpft sind, auf den in der Tabelle im TableView Portlet geklickt wurde. Wenn die Abfrage keine Ergebnisse zurückliefert, sollte eine Fehlermeldung z.B. durch eine Dialogbox angezeigt werden. Bei Ergebnissen, die mehrere VO-Views betreffen wird eine Dropdownliste angezeigt, über die diese ausgewählt werden können.

Details

Tabelle auswählen:

serviceID	ID	cpuAffinity	cpuClockSpeed	ramMemorySize	virtualMemorySize	osName	osVersion	connectivityIn	connectivityOut	clusterID	storageProvider	sourceURL	
realgrid- an.gwdg.de	10.1.1.17	2	2922	3500	8034	null	null	0	0	realgrid- an.gwdg.de	MD04	realgrid- an.gwdg.de	D T
realgrid- an.gwdg.de	10.1.1.20	2	2922	4020	8146	null	null	0	0	realgrid- an.gwdg.de	MD04	realgrid- an.gwdg.de	D T
realgrid- an.gwdg.de	10.255.255.100	0	2920	9000	24034	null	null	0	0	realgrid- an.gwdg.de	MD04	realgrid- an.gwdg.de	D T
realgrid- an.gwdg.de	10.255.255.101	0	2920	9000	24034	null	null	0	0	realgrid- an.gwdg.de	MD04	realgrid- an.gwdg.de	D T
realgrid- an.gwdg.de	10.255.255.102	0	2920	9000	24034	null	null	0	0	realgrid- an.gwdg.de	MD04	realgrid- an.gwdg.de	D T
realgrid- an.gwdg.de	10.255.255.103	0	2920	9000	17401	null	null	0	0	realgrid- an.gwdg.de	MD04	realgrid- an.gwdg.de	D T
realgrid- an.gwdg.de	10.255.255.104	0	2920	9000	24034	null	null	0	0	realgrid- an.gwdg.de	MD04	realgrid- an.gwdg.de	D T
realgrid- an.gwdg.de	10.255.255.105	0	2920	9000	24034	null	null	0	0	realgrid- an.gwdg.de	MD04	realgrid- an.gwdg.de	D T
realgrid- an.gwdg.de	10.255.255.106	0	2920	9000	24034	null	null	0	0	realgrid- an.gwdg.de	MD04	realgrid- an.gwdg.de	D T
realgrid- an.gwdg.de	10.255.255.107	0	2920	9000	24034	null	null	0	0	realgrid- an.gwdg.de	MD04	realgrid- an.gwdg.de	D T
realgrid- an.gwdg.de	10.255.255.108	0	2920	9000	24034	null	null	0	0	realgrid- an.gwdg.de	MD04	realgrid- an.gwdg.de	D T

214 Einträge gefunden, 11 Einträge werden angezeigt, von 1 bis 11, Seite 1 / 20

Figure 2

### Testszenerien:

- 13) Korrekte Darstellung des Portlets bei initialem Laden:
- Korrekte Darstellung der Daten aus der Datenbank in der Tabelle (korrekte VO-View, korrekte Anzahl der Datensätze und der Felder etc.)
  - Dropdownliste und Label für die Dropdownliste ausgeblendet
- Korrekte Anzeige der zu ResultSets aus der Tabelle AdminDomain. Dropdownliste auf disabled gesetzt.

Unit-Test:

Testklasse: test.unit.de.dgrid.dmon.portal.dao.TestDetailedViewDao

test.unit.de.dgrid.dmon.portal.util.TestDetailedColumnDataModelImpl

Getestete Klasse(n): de.dgrid.dmon.portal.dao.DetailedViewDao

de.dgrid.dmon.portal.dao.impl.DetailedViewDaoImpl

Modultest:

Testklasse: test.module.de.dgrid.dmon.portal.dao.TestDetailedViewDao

test.module.de.dgrid.dmon.portal.util.DetailedColumnDataModelImpl

Getestete Klasse(n): de.dgrid.dmon.portal.dao.DetailedViewDao

de.dgrid.dmon.portal.dao.impl.DetailedViewDaoImpl

Ergebnisse:

Unit- und Modultests wurden erfolgreich durchgeführt. Details zu den Testergebnissen können dem Testbericht im Anhang entnommen werden.

- 14) Korrektes Neuladen des Portlets bei Mausklick auf eines der ID-Felder im TableView Portlet:
- Korrektes Auslösen des Events
  - Korrektes (Neu)Setzen des selektierten Feldinhalts und Feldbezeichners

Durchführung: siehe TableViewPortlet, Testszenario 11

Ergebnisse: siehe TableViewPortlet, Testszenario 11

- 15) Korrektes Neuladen des Portlets bei Auswahl einer neuen VO-View in der Dropdownliste:
- Korrektes Neuladen der Daten
  - Korrektes Re-Rendern der View

Durchführung: Als Integrationstest mit JWebUnit/Selenium IDE

Testklasse: `test.integration.web.portlets.SeleniumTestDetailsTableChangeEvent`

Ergebnisse:

Funktionalität konnte erfolgreich getestet werden. Auswahl eines anderen Eintrags aus der Dropdownliste führte zum Neuladen der Daten in der Tabelle für das richtige übergeordnete Feld.

### **GMap Portlet:**

Das Portlet ist eine Java Applikation, die alle VO-Views anzeigt, die in irgendeiner Form inhaltlich mit dem Feldinhalt verknüpft sind, auf den in der Tabelle im TableView Portlet geklickt wurde. Wenn die Abfrage keine Ergebnisse zurückliefert, sollte eine Fehlermeldung z.B. durch eine Dialogbox angezeigt werden.

### **Testszenarien:**

- 16) Korrekte Darstellung der Daten / des Portlets beim initialen Laden:
- Prüfung der Korrektheit der SQL-Abfrage und der Daten, die auf diese zurückgeliefert wurden
  - Korrektes Setzen der Marker auf der Landkarte

Durchführung: Als Unit- und Modultest mit JUnit und Spring Unit Tests

Unit-Test:

Testklasse: `test.unit.de.dgrid.dmon.portal.dao.TestLocationDao`

Getestete Klasse(n): `de.dgrid.dmon.portal.dao.LocationDao`

`de.dgrid.dmon.portal.dao.impl.LocationDaoImpl`

Modultest:

Testklasse: `test.module.de.dgrid.dmon.portal.dao.TestLocationDao`

Getestete Klasse(n): `de.dgrid.dmon.portal.dao.LocationDao`

Ergebnisse:

Die Daten werden initial korrekt geladen und die Marker mit den korrekten Koordinaten gesetzt.

### 3.2.2 Sicherheit

Der Bereich Sicherheit kann als Teilbereich der Funktionalität gesehen werden, ist aber so wichtig, dass er gesondert geprüft werden sollte. Hierbei sollen vor allem das entworfene Rollenkonzept und die Funktionalität der VO-Awareness geprüft werden. Ebenso soll überprüft werden, wie sicher die gewählten Datenübertragungswege sind.

#### Testscenario 1: Rollenspezifische Filter

- 17) Je nachdem welche Rolle einem Benutzer zugeordnet wurde, wird die Sicht auf die VO-View zusätzlich auch rollenspezifisch gefiltert:
- Korrekte Funktionalität des Filters für die Rolle Nutzer einer VO
  - Korrekte Funktionalität des Filters für die Rolle VO-Repräsentant
  - Korrekte Funktionalität des Filters für die Rolle Zentrale Infrastrukturdienste
  - Korrekte Funktionalität des Filters für die Rolle D-Grid Operation Center

Durchführung: Als Integrationstest mit JWebUnit/Selenium IDE

Testklasse: test.integration.web.security.SeleniumTestRoleManagement

Ergebnisse:

Funktionalität konnte erfolgreich getestet werden. Alle Rollen wurden korrekt gesetzt, alle Werte korrekt gefiltert.

### 3.2.3 Darstellung

Für den Bereich Darstellung sollte vor allem die Gleichartigkeit der grafischen Darstellung in allen gängigen Browsertypen für die Benutzergruppe getestet werden. Hierfür soll sowohl die Gewährleistung der gleichen Funktionalität in allen Browsertypen geprüft werden als auch die Gleichartigkeit der Darstellung als Solches.

#### Testscenario 1: Browsertest

- 18) Grid-User greifen per Web auf das D-MON Portal zu um die Ihrer Rolle entsprechenden Monitoringdaten zu betrachten. Getestet wird die erfolgreiche Nutzung mit den Webbrowsern: Internet Explorer (verschiedene Versionen), Safari (aktuelle Release) und Mozilla Firefox (aktuelle Release):
- Gleichartigkeit der Darstellung in allen Browsern

- Korrekte Funktionalität aller Portlets in allen Browsern

Durchführung:

Test von Hand, Aufruf der Seite in allen gängigen Browsern

Ergebnis:

Bis heute stellen alle gängigen Browser, außer Safari die Portalseiten korrekt dar (siehe Abb. 6 -8).

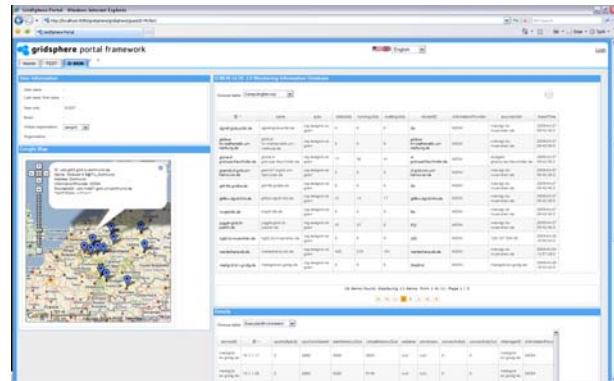


Figure 3: Internet Explorer 7

An der korrekten Funktionalität für Safari wird momentan noch gearbeitet. Bis dato existieren noch Probleme mit der Inter-Portlet Communication, die auf den internen JavaScript Engine von Safari zurückzuführen sind. Dadurch entstehen Probleme beim dynamischen Nachladen bei User-Events.

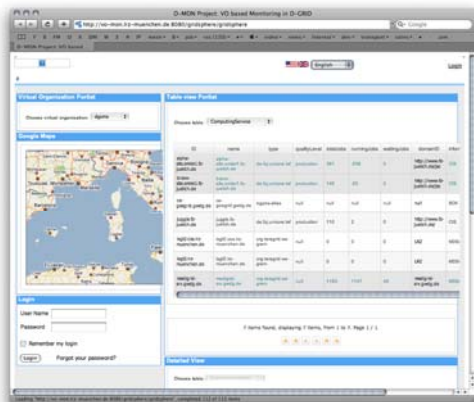


Figure 7: Firefox

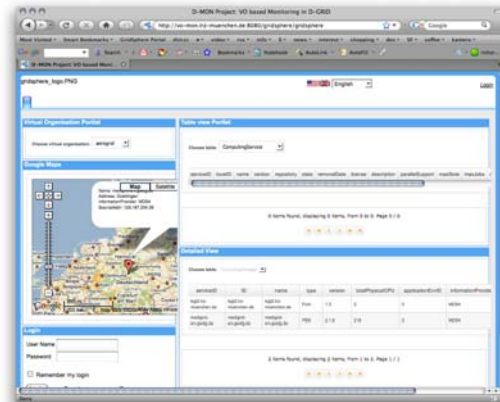


Figure 4: Safari 3.2

### 3.2.4 Benutzerfreundlichkeit

Die Tests in Bezug auf Benutzerfreundlichkeit einer Anwendung beziehen sich sowohl auf den Bereich Installation und Deployment als auch auf die Benutzung der grafischen Benutzeroberfläche an sich.

#### Testszenario 1

- Ein GoC möchte einen Gridsphere-Server anbieten, welcher über die D-MON Portlets verfügt. Getestet werden:
  - Installation der Software und Schaffen der Voraussetzungen für die Installation
  - Lauffähigkeit des Servers mit den Portlets
  - Möglichkeit zum Login eines Benutzers

Installationsvoraussetzungen:

- Apache Ant 1.7 oder hoehere Version
- SVN Client
- Java v 1.5 oder hoehere Version
- Apache Tomcat 5.5 oder hoehere Version
- Gridsphere 3.0.x oder hoehere Version

Setzen der Umgebungsvariablen:

- JAVA\_HOME: Java Installation directory
- CATALINA\_HOME: Tomcat Installation directory
- ANT\_HOME: Ant Installation directory

1) Hinzufuegen des Attributs emptySessionPath zum Connector Tag in der Datei server.xml:

- Oeffnen Sie die Datei \$CATALINA\_HOME/conf/server.xml ...

- und aendern Sie den Eintrag:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
    <Connector port="8080" maxHttpHeaderSize="8192" ...
```

... zu:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
    <Connector port="8080" maxHttpHeaderSize="8192"
emptySessionPath="true" ...
```

2) SVN Checkout der D-MON Web-Applikation nach [gridsphere\_home]/projects

3) Oeffnen Sie Ihre DosBox/Shell. Navigieren Sie zum D-MON Ordner und fuehren Sie das Kommando 'ant install' aus, um den Installationsprozess zu starten.

4) Nachdem die Installation abgeschlossen wurde, oeffnen Sie das Gridsphere Portal in ihrem Browser. Insofern ihr Webserver waehrend des Installationsprozesses lief, sollten Sie ihn nun neustarten. Loggen Sie sich nun als Admin ein und kontrollieren unter Administration->Portlets, ob die Installation erfolgreich war.

5) Gehen Sie nun zum Bereich Administration->Roles und legen Sie die folgenden Rollen an:

- REPRESENTATIVE fuer den VO-Repraesentanten
- GOC fuer das D-GRID Operation Center
- SERVICE fuer den Zentralen Infratstrukturdienst
- VOUSER fuer den Nutzer einer VO

## 4 Performanz-Tests

### 4.1 Lasttests

#### 4.1.1 Einleitung

Wenn die entwickelten Komponenten und das System in einem funktional stabilen Zustand sind, muss untersucht werden, wie die erwartete Last das Verhalten des Systems beeinträchtigt. Mit dem Lasttest wird eine hohe Systemlast erzeugt und es werden die Zeiten gemessen, die die Adapter, das Frontend und die Datenbank benötigt, um ihre Funktion zu erfüllen. Durch diese Tests wird die Zuverlässigkeit des Systems geprüft. Mit Hilfe der Performancemessungen werden Informationen über das D-MON System gesammelt mit denen es möglich ist, Leistungsengpässe zu erkennen.

#### 4.1.2 Testscenarien:

##### **Testscenario 1: Middleware-übergreifende Performance-Tests des Backends:**

20) Mit Hilfe der entwickelten D-MON Adaptern werden die von D-MON unterstützten Informationsdienste abgefragt und die Monitoring Daten werden in das gewünschte Schema umgewandelt. Verschiedene Adaptor-Implementierungen können durch diese Performancemessungen getestet werden, um zu entscheiden, welche Implementation das Systemverhalten verschlechtert oder verbessert. Die Durchführung von Messungen und die Performanceanalyse müssen die möglichen Engpässen der einzelnen Module und Units identifizieren.

*Durchführung:*

*Verschiedene D-MON Adaptern mit aufgeteilten- und zusammengeführten Transformermodellen wurden prototypisch implementiert. Bei einem aufgeteilten Adapter ist der Transformer aufgeteilt (mehrere XSLTs mit einzelnen Schema-Teilchen) und ein Teil transformiert nur einen Teil des Schemas. Nachdem der Informationsdienst abgefragt wird, wird immer eine Teilmenge der Monitoring Daten in das gewünschte Schema umgewandelt. Damit wird weniger Last erzeugt, muss aber mehrere Umwandlungsiteration gemacht werden.*

*Der zusammengeführte Adapter benutzt nur einen Transformer, wo alle XSLTs zusammengefügt sind. Nachdem der Informationsdienst abgefragt wird, werden in einem Lauf die Monitoring Daten komplett in das gewünschte Schema umgewandelt.*

*Es wurden mehrere Datenquellen, darunter auch ein Site-BDII und der Toplevel-BDII des D-Grid mit den unterschiedlichen Adaptorimplementationen abgefragt.*

*Ergebnisse:*

*Die gesamte Übersetzungszeit mit dem aufgeteiltem Adapter ist zweimal so gross, wie die Umwandlungszeit mit dem zusammengeführten Adaptor obwohl der gleiche Anzahl der SQL-Befehle erzeugt wird. Es wird nicht wesentlich grösseren Last während der Umwandlung mit*

dem zusammengefügtem Transformer erzeugt. Aus diesen Gründen wurde für die D-MON Architektur das zusammengefügte Transformermodell gewählt.

### **Testscenario 2: Datenbank-UPDATE Lasttests:**

21) Die in das GLUE 2.0 Schema konvertierten Monitoring-Daten sollen durch Simulation von vielen parallelen Adaptoren in die D-MON Datenbank eingefügt werden. Diese Lasttests sollen über einen längeren Zeitraum laufen. Während der Tests werden die Zeiten gemessen, die das System für Einfügen der Monitoring-Daten benötigt.

*Durchführung:*

*Verschiedene D-MON Adaptoren, die das zusammengefügte Transformer- und Schemamodell implementiert haben, wurden parallel gestartet. Die Adaptoren haben vorher gespeicherte und teilweise simulierte Monitoring-Daten aus den drei verschiedenen Middleware-Instanzen in die D-MON Datenbank geschrieben.*

*Dieser Test wurde mehrmals, jeweils über einen Zeitraum von 3 Stunden durchgeführt.*

*Ergebnisse:*

*Es wurden alle Einträge in die Datenbank korrekt eingefügt.*

*Es wurden alle Einträge in die Datenbank korrekt eingefügt. Beim Dateneinfügen traten keine grösseren Verzögerungen auf."*

### **UPDATE test**

<b>Test</b>	<b>Description</b>	<b>Successful Tests</b>	<b>Errors</b>	<b>Mean Time</b>	<b>Mean Time Standard Deviation</b>
Test 1	DB Update MDS4	10000	0	421	1409
Test 2	DB Update CIS	10000	0	240	877
Test 3	DB Update BDII	10000	0	303	2010

### **Testscenario 3: Datenbank-Abfrage Lasttests:**

22) Die D-MON Datenbank mit den GLUE 2.0 Monitoring Daten soll durch Simulation vieler paralleler Klienten abgefragt werden. Diese Lasttests sollen über einen längeren Zeitraum laufen. Während der Tests werden die Zeiten gemessen, die das System für das Ausführen der Abfragen benötigt.

*Durchführung:*

*Verschiedene Klienten wurden parallel gestartet. Die Klienten haben unterschiedliche Rollen simuliert und von der Rolle abhängig verschiedene Teile des GLUE 2.0 Schemas abgefragt.*

*Dieser Test wurde mehrmals, jeweils über einen Zeitraum von 3 Stunden/Tagen durchgeführt.*

*Ergebnisse:*

*Die Abfragen wurden korrekt beantwortet. Die Abfragen wurden korrekt beantwortet. Verzögerungen beim Datenabfragen traten nicht auf."*

## QUERY test

Test	Description	Successful Tests	Errors	Mean Time	Mean Time Standard Deviation
Test 1	DB VO Portlet	40000	0	24.3	16.3
Test 2	DB GoogleMap Portlet	40000	0	38.2	18.2
Test 3	DB GLUE2 Dropdown Portlet	40000	0	24.6	18.6
Test 4	DB GLUE2 ComputingServiceVO	40000	0	37.1	27.9
Test 5	DB Details Portlet	40000	0	31.4	16.7
Test 6	DB GLUE2 ComputingShareVO	40000	0	102	36.4
Test 7	DB GLUE2 ComputingManagerVO	40000	0	50.6	26.4

### Testscenario 4: Lasttests des Frontends:

23) Durch Simulation vieler paralleler Benutzer oder durch Wiederholen von Aktionen und Abfragen eines Nutzers wird Last auf dem Frontend erzeugt. Diese Tests sollen über einen längeren Zeitraum laufen. Während dieser Lasttests werden die Zeiten gemessen, die das D-MON System für Benutzeraktionen benötigt.

Die Lasttests des Frontends wurden online mit einem Testzugang bei BrowserMob ([www.browsermob.com](http://www.browsermob.com)) durchgeführt. BrowserMob führt parallel ein Selenium Skript aus und simuliert damit den parallelen Zugriff von bis zu 50 Seitenaufrufe auf einmal.

Durch die hohe Anzahl an gleichzeitigen Zugriffen auf die Datenbank bzw. die Abfrage des OGSA-DAI Services (und die durch OGSA-DAI generierten langsamen Antwortzeiten) fuer den Aufbau der Portalseite, konnten die Lasttests nur fuer bis zu 10 gleichzeitige Seitenaufrufe erfolgreich durchgeführt werden. Ansonsten kam es zu Problemen mit dem Java Heap Space.



Figure 5: Throughput - Bytes received (total and by script)

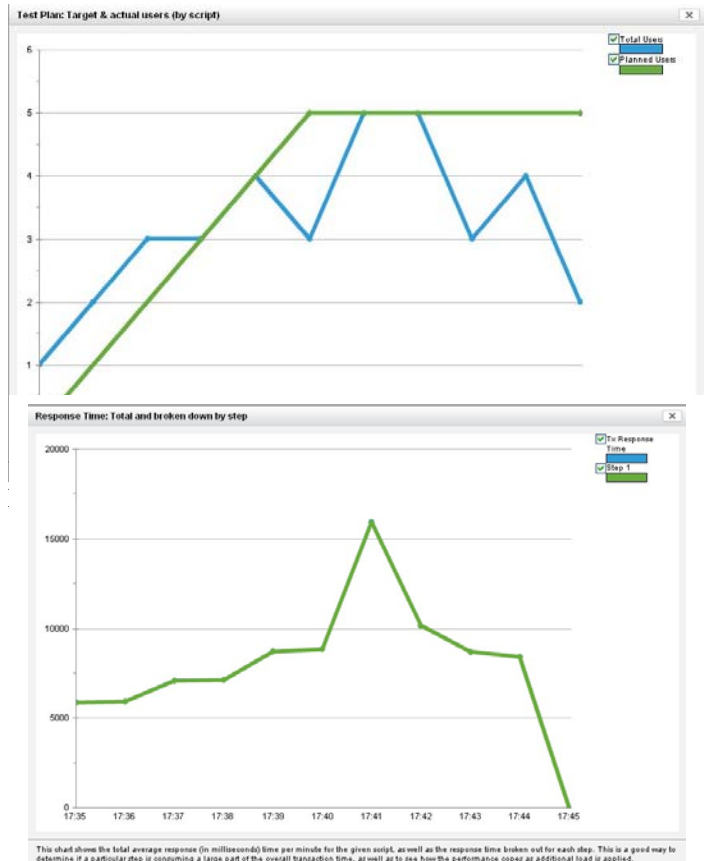


Figure 6: Response time - Total and broken down by step

### 4.1.3 Eingesetzte Tools

Modul- und Unittesttools (s. Oben) und Eigenentwicklungen

## 4.2 Verteilungstests

Verschiedene Möglichkeiten der Verteilung der Daten müssen bezüglich ihrer Funktionalität, Stabilität und Skalierung untersucht werden. Getestet werden sollen unter anderem:

Zentraler Ansatz

Alle Adapter fügen ihre Daten in eine Zentrale Datenbank Instanz ein.

DB-Cluster

Alle Adapter fügen die Daten in eine Site-lokale Datenbank ein, die Bestandteil eines Grid weiten Datenbank Clusters ist.

DB-Verbünde

Alle Adapter fügen die Daten in eine Site-lokale Datenbank ein. Daneben existiert eine/ bzw. mehrere Datenbank(en), die die Informationen der Site-lokalen Datenbanken abfragen und vereinigen kann/können.

Dies sind die drei grundsätzlich möglichen Modelle zur Datenverteilung. Alle Modelle wurden prototypisch implementiert und hinsichtlich der genannten Kriterien getestet, um einen stabilen Produktivbetrieb gewährleisten zu können. Der zentrale Ansatz ist der technisch einfachste und wurde daher als erstes implementiert. Er zeigt während der gesamten Entwicklungs- und Testphase nie Probleme bezüglich Stabilität und Performanz, obwohl alle Daten der D-Grid Monitoringsysteme eingespeist wurden. Die eigentlich konzeptionell stabilere DB-Cluster Lösung erwies sich als unbrauchbar, da die Performanz des über ganz Deutschland verteilten Datenbank-Clusters trotz Gigabit-Verbindung um ca. den Faktor 10 einbrach. SQL-Einfüge-Operationen die im zentralen Ansatz nur wenige Sekunden dauerten, benötigten im geclusterten Aufbau 30 Sekunden und länger. Hinzu kamen Ausfälle einzelner Knoten, die zum Teil das gesamte Cluster außer Betrieb setzten. Auch ein Update auf eine aktuellere MySQL Version half nicht. Der Cluster-Ansatz lässt sich damit in der Praxis nicht realisieren.

Ein Datenbank-Verbund aus lokalen Datenbanken der Sites, die über eine bzw. mehrere zentrale Datenbanken via der federated Engine zu einer Gesamtansicht vereinigt werden, funktionierte prinzipiell, wenn auch mit leichten Performanzeinbußen. Das eigentliche Problem dieses Lösungsansatzes bestand im Ausfall einer lokalen Datenbank, die eine Abfrage der globalen DB verhinderte. Dies lag daran, dass die Abfrage einer entfernten, nicht verfügbaren Datenbank die gesamte Abfrage verhinderte. Ein Abfangen dieses Fehlers war nicht möglich, so dass auch dieser Ansatz praktisch nicht realisierbar erscheint.

Aus diesen Gründen wurde für die D-MON Architektur ein zentraler Ansatz gewählt. Hochverfügbarkeit und Skalierung des Systems muss außerhalb der Datenbank sichergestellt werden. Da alle zentralen Instanzen am LRZ auf einem Virtualisierungs-Cluster betrieben werden, ist es jedoch kein Problem, die Hardware der Systeme zu skalieren und Hochverfügbarkeit zu gewährleisten.

## **5 Zusammenfassung**

Die beschriebenen Tests beinhalten die Funktionalitätsprüfung für den Benutzer der zur Verfügung gestellten Kommandozeilen- und Web-Clients sowie auf der Serverseite die Funktionsprüfung der Adapter und Module, die an jeder D-Grid Site installiert werden können. Letztere ergänzen und unterstützen die Realisierung von site- und middleware-übergreifenden Informationsdiensten.

Zweck, Funktionsweise und Spezifikation der Tests sind angepasst an die in AP1.3. [1] spezifizierten Neu- und Weiterentwicklungen des Monitoringsystems. Um die Interoperabilität der entwickelten Komponenten und die Integration mit dem VO-Management zu testen, soll der Schwerpunkt auf dem Zusammenspiel der Informationssysteme und Adapter der drei eingesetzten Middlewares und der Einbindung des VO-Managements liegen. Insbesondere wird auch die Nutzeroberfläche ausgiebig untersucht.

## 6 Referenzen

- [1] Softwaretechnische Analyse und Design, D-MON Project Deliverable, Oktober 2008  
[http://www.d-grid.de/fileadmin/user\\_upload/documents/D-MON/Design-AP1.3.final.pdf](http://www.d-grid.de/fileadmin/user_upload/documents/D-MON/Design-AP1.3.final.pdf)
- [2] Spezifikation Middlewareübergreifender Monitoring und VO-Integrations Tests, D-MON Projekt Deliverable, Dezember 2008,  
[http://www.d-grid.de/fileadmin/user\\_upload/documents/D-MON/AP2\\_4.pdf](http://www.d-grid.de/fileadmin/user_upload/documents/D-MON/AP2_4.pdf)
- [3] Frank Cohen, Java Testing an Design: From unit testing to automated web test. Prentice Hall PTR, Upper Saddle River (2004)
- [4] Sammlung von Anforderungen, D-MON Projekt Deliverable, Januar 2008  
[http://www.d-grid.de/fileadmin/user\\_upload/documents/D-MON/Anforderungsanalyse-Version1.0.pdf](http://www.d-grid.de/fileadmin/user_upload/documents/D-MON/Anforderungsanalyse-Version1.0.pdf)