



# **Spezifikation Middleware- übergreifender Monitoring und VO-Integrations Tests für AP 2.4**

**Anne Milbert (AEI)**  
**Gevorg Poghosyan (FZK/KIT)**  
**Tibor Kálmán (GWDG)**  
**Tobias Lindinger (LMU)**  
**Timo Baur (LRZ)**  
**Mathilde Romberg (FZJ)**

5. Dezember 2008

## **INHALTSVERZEICHNIS**

<b>1</b>	<b>EINLEITUNG</b> .....	<b>3</b>
<b>2</b>	<b>MIDDLEWARE-ÜBERGREIFENDE TESTS</b> .....	<b>5</b>
2.1	TESTSZENARIEN FÜR ADAPTER .....	5
2.2	TESTSZENARIEN FÜR DIE DATENBANK .....	6
<b>3</b>	<b>VO-BASIERTE- UND INTEGRATIONSTESTS</b> .....	<b>8</b>
3.1	TESTSZENARIEN FÜR DIE DATENBANK .....	8
3.2	TESTSZENARIEN FÜR DAS FRONTEND.....	8
<b>4</b>	<b>PERFORMANZ-TESTS</b> .....	<b>14</b>
4.1	LASTTESTS .....	14
4.2	VERTEILUNGSTESTS .....	15
<b>5</b>	<b>ZUSAMMENFASSUNG</b> .....	<b>16</b>
<b>6</b>	<b>REFERENZEN</b> .....	<b>17</b>

## 1 Einleitung

Das D-MON Projekt entwickelt ein middleware-übergreifendes, VO-basiertes Monitoring System, dessen Entwurf in [1] beschrieben ist. Im vorliegenden Dokument werden Anforderungen für die durchzuführenden Tests des erstellten Systems gesammelt.

In Anlehnung an die beiden Hauptstrukturelemente des Projektes sollen die Tests die Funktionalität eines *middleware-übergreifenden* und die eines *VO-spezifischen Monitorings* abdecken. Der Fokus soll ferner auf die Hauptfaktoren Funktionalität der Anwendung und ihrer einzelnen Komponenten sowie Performanz gerichtet werden. Hierbei sollen zuerst alle Module für sich und dann insgesamt im Zusammenspiel miteinander getestet werden, um die Interoperabilität der Komponenten und deren Integration mit dem VO-Management feststellen zu können. Hierzu werden in den folgenden Abschnitten verschiedene Testszenarien entwickelt, die in der nächsten Projektphase realisiert werden sollen. Analog zu den Testszenarien müssen vor Beginn der Tests Ziele vereinbart werden und Testanforderungen festgelegt werden. Diese sollen die angestrebten Ergebnisse der Tests beinhalten und die Anforderungen an die entsprechenden Tests definieren.

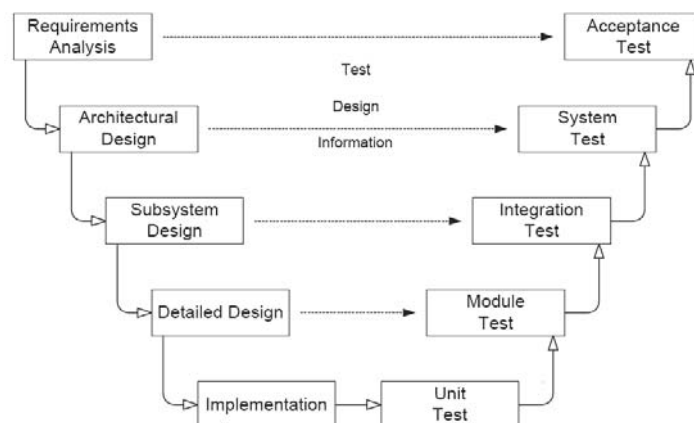


Abb. 1: Zuordnung Test- zu Entwicklungsphase

Wie Abbildung 1 anschaulich am V-Modell [2] erläutert, werden in der Softwareentwicklung die verschiedenen Entwicklungsphasen bestimmten Testphasen zugeordnet. In der Regel sollten Entwicklungs- und Testphase zeitlich überlappend ablaufen.

In der Grafik werden die Testphasen von oben nach unten immer feingranularer in Bezug auf den Bereich der Software, der durch sie getestet wird. Unit- und Modultest können auf Ebene der Module unabhängig voneinander durchgeführt werden. Durch Integrations-, System- und Benutzerakzeptanztests wird das Zusammenspiel der entwickelten Module miteinander geprüft. Sie setzen deswegen erfolgreich durchgeführte Modul- und Unittests voraus.

Modultests testen die Funktionsfähigkeit von Modulen. Als Modul kann in diesem Kontext ein eigenständiger Teil des Gesamtsystems verstanden werden. Hierbei kann ein Modul auch wieder als eigenes Gesamtsystem auftreten, das wiederum eigene Module enthält.

Unittests setzen an der kleinstmöglichen Einheit an, in die ein Modul zerlegt werden kann. Da diese von Modul zu Modul sehr unterschiedlich sein können, wird in jedem der folgenden Abschnitte über die Tests der Module die Definition des Begriffs Unit in dem speziellen Kontext vorausgehen.

Ein Integrationstest testet das Zusammenspiel der Module miteinander. Er setzt, wie bereits oben erwähnt, erfolgreich durchgeführte Unit- und Modultests voraus. Der Systemtest besteht aus einer Reihe von Einzeltests, die das gesamte System testen. Erst wenn dieser erfolgreich abgeschlossen ist, sollte ein Benutzerakzeptanztest durchgeführt werden. Hier wird die Anwendung auf Kriterien wie intuitive Bedienung, Gesamtfunktionalität, Reaktion des Systems im Fehlerfall etc. getestet werden

Die hier zu testende Applikation lässt sich in drei Hauptmodule aufteilen: das Backend, die Datenbank und das Frontend. Diese bilden die Gesamtanwendung und ihr reibungsloses Zusammenspiel kann nur durch ihre eigene reibungslose Grundfunktionalität gewährleistet werden. Die Tests sollen deswegen vom kleinstmöglichen Element eines Moduls (Unit) hin zur Gesamtanwendung gehen und zwar begleitend zur Entwicklung, um schnellstmöglich auf fehlerhafte Annahmen in Logik, Implementierung oder Design reagieren zu können.

Die Schwerpunkte der Testszenarien sind middleware-übergreifendes Monitoring und VO-basiertes Monitoring. Hierbei sind insbesondere eine Überprüfung des Zusammenspiels aller drei eingesetzten Middlewares und die Einbindung des VO-Managements in das Informationssystem notwendig.

Dieses Deliverable ist entsprechend des beschriebenen Vorgehens strukturiert: Kapitel 2 behandelt die Tests zum middleware-übergreifenden Monitoring und ist nach Testszenarien für Adapter und Datenbank unterstrukturiert. Analog enthält Kapitel 3 die zum VO-basierten Monitoring gehörenden Tests mit den Unterpunkten Datenbank und Frontend sowie Integrationstests. Übergeordnete Performanz-Tests werden in Kapitel 4 behandelt.

## 2 Middleware-übergreifende Tests

### 2.1 Testszzenarien 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

Wie bereits erläutert werden die Modultests vorerst analog zu den drei unterstützten Middlewares aufgegliedert. Auf dieser Ebene werden dann für jedes Modul die Testszzenarien für die Unit Tests entwickelt. Erzielen alle Unit Tests für ein Modul die erwarteten Testergebnisse, gilt der Test des Moduls als erfolgreich.

Die entwickelten Testszzenarien sollten möglichst eng an den eigentlichen Funktionalitäten der Anwendung orientiert sein, welche für alle drei Module zunächst gleiche Szenarien abdecken muss:

Im Folgenden werden weitere, spezifische Szenarien für die einzelnen Module festgelegt.

#### Testszzenarien

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

<u>Middleware</u>	<u>Informationsdienst</u>	<u>Output</u>	<u>Abfragetool</u>	<u>Output</u>
Glite ab v.3.0	LDAP/BDII server	LDIF ( <i>Glue Schema v 1.2</i> )	Perl Script ldap2xml.pl ( <i>eigenentwicklung</i> ) oder java class org.dsmltools.LDAP2DSML	XML ( <i>DSML Schema</i> )
Globus Toolkit ab v.4.0	MDS4 server	XML ( <i>Glue Schema v.1.1</i> )	WSRF client	XML ( <i>Glue Schema v.1.1</i> )
Unicore ab v.6.1.3	CIS server	XML ( <i>GLUE Schema V2.0</i> )	unicore commandline client	XML ( <i>Glue Schema v.2.0 release 32</i> )

- 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.

- 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 20 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.

### 2.1.2 Eingesetzte Tools

- **XSLTUnit**: Unit testing framework für XSLTs.  
URL: <http://xsltunit.org/>
- **LDAP Perl Klient ldapxml**  
URL: <http://ldap.perl.org/>
- **Java LDAP2DSML tool**  
URL: <http://www.dsmltools.org/>
- **wsrf-query**: Globus Toolkit 4 Java WS Core Command Line Client  
URL: <http://www.globus.org/toolkit/docs/4.0/common/javawscore>
- **ucc**: UNICORE command client  
URL: <http://www.unicore.eu>

## 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 Einfügeversuches durch die Datenbank abgelehnt. Des Weiteren werden feldspezifisch einige Prüfungen auf bestimmte Formate und Typen durchgeführt. Geprüft werden muss daher nur, ob die entsprechenden Daten aus den Middleware spezifischen Monitoring Systemen konsistent in das GLUE 2.0 Format konvertiert wurden 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.

### **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.

### **3 Vo-basierte- und Integrationstests**

#### **3.1 Testsznarien 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**

#### **Testsznario 1: Anbindung der VO-Management DBs**

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

#### **Testsznario 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.
- 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 muss im Rahmen der Tests ebenfalls getestet werden, indem Einträge der Adapter nicht aktualisiert werden und Berechtigungen im GRRS verändert werden. Das Ergebnis muss manuell überprüft werden.

#### **Testsznario 3: CHECK**

- 9) Die Daten der Informationsdienste müssen einheitlich und korrekt in das GLUE 2.0 Schema konvertiert und in die Datenbank eingefügt werden. VO-spezifischen Views sollen verifiziert werden.

#### **Testsznario 4: Generieren von VO-spezifischen Sichten**

- 10) 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.

#### **3.2 Testsznarien für das Frontend**

Die Bereiche, in denen das Frontend der D-MON Gesamtapplikation getestet werden muss, lassen sich klar voneinander abgrenzen. Als erster und gleichzeitig auch wichtigster Bereich muss hier die Funktionalität genannt werden. Gemeint ist damit, die Aktualität und Validität der

ausgegebenen Daten. Thematisch gehören hierzu unter anderem die Gewährleistung der reibungslosen Funktionalität des VO-spezifischen Zugriffs auf den D-MON-Dienst, die Datenbank, und die Web-Portlets.

Des Weiteren muss der Bereich Sicherheit getestet werden. Die Funktionalität der rollenspezifischen Filter muss hier besondere Aufmerksamkeit geschenkt werden. Ebenso sollte die Sicherheit der Verbindung zwischen Datenbank und Frontend überprüft werden.

Als ein weiterer Bereich, der intensiv getestet werden sollte lässt sich die Darstellung identifizieren. Hier muss vor allem über Tests mit verschiedenen Browsertypen gewährleistet werden, dass die Anwendung in jedem der Browser gleich aussieht und alle Portlets funktionieren.

Im Bereich zwischen Darstellung und Funktionalität ist die Inter-Portlet Communication (IPC) angesiedelt. Getestet werden muss in diesem Kontext, ob das dynamische Nachladen von Informationen in einem Portlet durch ein User Event in einem anderen Portlet getriggert werden kann. Ebenfalls getestet werden muss, ob die Anwendung immer noch reibungslos funktioniert, wenn der Endnutzer die Verwendung von JavaScript in seinem Browser deaktiviert hat. In den folgenden Abschnitten soll dies nun genauer erläutert und verschiedene Testszenarien für diese Punkte entwickeln werden.

Als weiterer wichtiger Punkt für die Tests ist die Benutzerfreundlichkeit zu sehen. Hierfür sollte getestet werden, ob die Installation des Frontends durch einen ungeschulten Benutzer durchführbar ist. Ebenso sollte die Benutzung der Seite durch Tests mit potentiellen Endnutzern auf intuitive Bedienbarkeit und eingängige Benutzerführung durchgeführt werden.

### **3.2.1 Funktionalität**

Die Tests im Bereich der Funktionalität beziehen sich hauptsächlich auf die Validität der Informationen/Daten, die durch das Frontend ausgegeben werden. Hier soll jedes einzelne Portlet in speziell auf seine Funktionalität zugeschnittenen Testszenarien überprüft werden. Folgende Testszenarien wurden hierfür Bereich entwickelt:

#### **TableViewPortlet:**

Das Portlet ist eine Java Applikation, die in einer Dropdownliste alle in der D-MON Datenbank enthaltenen VO-Views anzeigt und einer Tabelle die diese VO-Views anzeigen kann. Defaultmässig wird die VO-View ComputingService\_VO angezeigt.

#### **Testszenarien:**

- 11) Korrekte Darstellung der Daten und der VO-Views beim initialen Laden der Seite:
  - Korrekte Anzeige aller möglichen VO-Views
  - Korrekte Anzeige der Daten aus der VO-View ComputingService\_VO
  - Korrekt gesetzte VO
  
- 12) Korrekte 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
- 13) Per Mausclick 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:
- Mausclick auf ein Primär- oder Fremdschlüsselfeld muss ein Event auslösen
  - Feldname und –inhalt des geklickten Feldes müssen korrekt übertragen werden
- 14) Überprüfung, ob die Funktionalität den in Dokument AP 1.1 definierten Anforderungen entspricht.

#### **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 zB. durch eine Dialogbox angezeigt werden. Bei Ergebnissen, die mehrere VO-Views betreffen wird eine Dropdownliste angezeigt, über die diese ausgewählt werden können.

#### **Testszzenarien:**

- 15) 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
- 16) Korrektes Neuladen des Portlets bei Mausclick auf eines der ID-Felder im TableView Portlet:
- Korrektes Auslösen des Events
  - Korrektes (Neu)Setzen des selektierten Feldinhalts und Feldbezeichners
- 17) Korrektes Neuladen des Portlets bei Auswahl einer neuen VO-View in der Dropdownliste:
- Korrektes Neuladen der Daten
  - Korrektes Re-Rendern der View
- 18) Überprüfung, ob die Funktionalität den in Dokument AP 1.1 [3] definierten Anforderungen entspricht.

#### **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. Bei Ergebnissen, die mehrere VO-Views betreffen wird eine Dropdownliste angezeigt, über die diese ausgewählt werden können.

#### **Testszzenarien:**

- 19) 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
  
- 20) Auslösen eines Events bei Mausklick auf einen Marker:
  - Prüfung, ob das Event ausgelöst wird
  - Prüfen, ob die korrekten Daten mit dem Event übertragen werden
  
- 21) Überprüfung, ob die Funktionalität den in Dokument AP 1.1 [3] definierten Anforderungen entspricht.

### **3.2.2 Sicherheit**

Der Bereich Sicherheit kann als Teilbereich des Bereichs 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.

#### **Testszzenario 1: VO Rechtekonzept**

- 22) Nach Authentifizierung eines Nutzers bei der Grid Authentication und Authorization Infrastructure (AAI) soll dessen VO ermittelt werden und er darf nur entsprechend dieser Zugriffsrechte auf die ihm zugeordneten spezifischen Sichten zugreifen können:
  - Korrekte Authentifizierung
  - Korrekt zurückgelieferte VO(s)
  - Korrekte Funktionalität des VO-Filters

#### **Testszzenario 2: Rollenspezifische Filter**

- 23) 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 Infrastruktur-dienste
  - Korrekte Funktionalität des Filters für die Rolle D-Grid Operation Center

#### **Testszzenario 3:**

- 24) Überprüfung, ob die Funktionalität den in Dokument AP 1.1 [3] definierten Anforderungen entspricht.

### **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.

#### **TestszENARIO 1: Browsertest**

25) 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

### **3.2.4 Inter-Portlet Communication**

Normalerweise muss die gesamte Portalseite neu geladen werden, wenn eines der Portlets neu geladen werden muss. Durch die Implementierung der IPC in Kombination mit Ajax ist es jetzt möglich den Inhalt nur eines Portlets dynamisch neu zu laden, während der Rest der Portalseite davon unberührt bleibt. Es gilt in diesem Zusammenhang zu testen, ob bei Eintreten eines Events, das IPC auslöst alles korrekt abläuft.

#### **TestszENARIO 1: VO Auswahl**

26) Nach Auswahl einer VO sollen VO-spezifische Sichten angezeigt werden. Durch Auswahl einer neuen VO soll ein Event ausgelöst werden, dass das Neuladen der Daten in den anderen Portlets triggert:

- Korrektes Auslösen des Events
- Korrekte Funktionalität des Neu-Renderns der Sichten auf die Daten mit korrekt gesetzter VO

#### **TestszENARIO 2: Feldauswahl**

27) Per Mausklick auf eines der ID-Felder in der Tabelle des TableView Portlets sollen im DetailedView Porlet die Daten dynamisch entsprechend des ausgewählten Feldes neugeladen werden:

- Korrektes Auslösen des Events
- Korrekte Übertragung von Feldname und –inhalt.
- Korrektes Neuladen der Daten
- Korrektes Neuladen der View

### 3.2.5 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

28) 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

#### Testszenario 2

29) Ein VO Nutzer möchte das Frontend erstmals benutzen. Getestet werden sollen:

- Intuitive Bedienbarkeit der Benutzeroberfläche
- Nachvollziehbarkeit der Benutzerführung
- Verständnis der verwendeten Begrifflichkeiten

### 3.2.6 Eingesetzte Tools

- Clover: Code Coverage Analysis Tool, misst die Abdeckung der Unit Tests im Source Code  
URL: <http://www.atlassian.com/software/clover/>
- JUnit: Unit Testing Framework für Java Applikationen  
URL: <http://www.junit.org/>
- JMock: Java Framework zur Entwicklung von Mock Objekten, die die durchgeführten Unit Tests unabhängig von anderen Modulen machen  
URL: <http://www.jmock.org/>
- OGSA-DAI: Webservice basierte Schnittstelle zur Datenbank  
URL: <http://www.ogsadai.org/>

## **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 Testszenarios:**

##### **Testscenario 1: Middleware-übergreifende Performancetests des Backends:**

- 30) Mit Hilfe der entwickelten D-MON Adaptoren 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.

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

- 31) 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.

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

- 32) 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.

##### **Testscenario 4: Lasttests des Frontends:**

- 33) 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.

#### **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 ein 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 müssen hinsichtlich der genannten Kriterien getestet werden, um einen stabilen Produktivbetrieb gewährleisten zu können. Dazu wird ein entsprechender Prototyp instanziiert und einige Anfragen gestellt bzw. Monitoringinformation mit Hilfe der Adapter eingefügt. Die Dauer der einzelnen Aktionen wird gemessen. Im verteilten Fall werden zusätzlich einige Knoten deaktiviert um die Lauffähigkeit des Systems im Fehlerfall zu simulieren.

## **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] Frank Cohen, Java Testing an Design: From unit testing to automated web test. Prentice Hall PTR, Upper Saddle River (2004)
- [3] 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)