TextGrid
Virtuelle Forschungsumgebung
für die Geisteswissenschaften

Dokumentation Funktionale Tests (R 5.2.1)

Version 09.06.2015
Arbeitspaket 5.2
verantwortlicher Partner Hochschule Worms
Zentrum für Technologietransfer
und Telekommunikation (ZTT)

TextGrid

Virtuelle Forschungsumgebung für die Geisteswissenschaften

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Projekt: TextGrid – Institutionalisierung einer Virtuellen Forschungsumgebung in den Geisteswissenschaften
BMBF Förderkennzeichen: 01UG1203A
Laufzeit: Juni 2012 bis Mai 2015

Dokumentstatus: final
Verfügbarkeit: öffentlich
Autoren:

Küster, Marc Wilhelm
Selig, Thomas
Stuck, Christian

Revisionsverlauf:

Datum	Autor	Kommentare

Inhaltsverzeichnis

1. Technische Grundlagen

Die Hochschule Worms hatte bereits im Projekt-Abschnitt 2 des TextGrid-Projektes die funktionalen Tests übernommen. Diese wurden im dritten Projektabschnitt ausgebaut und an aktuelle Erfordernisse angepasst.
Die funktionalen Tests basieren auf dem Open-Source-Framework Sikuli[1]. Tests mit diesem Framework arbeiten mit der Methode des Screen-Capturing. Dabei wird der auf dem Bildschirm dargestellte Inhalt eingelesen und mit Bildverarbeitungsmethoden ausgewertet. Dieses Vorgehen hat gegenüber anderen Frameworks wie SWT-Bot[2] den Vorteil, dass die Testsoftware nicht direkt in der zu testenden Software integriert wird. Es wird getestet, was der Nutzer auf dem Bildschirm sieht und nicht, welche Werte ausgewählte Programmvariablen zu bestimmten Zeiten besitzen. Damit wird auch der große Bereich der Darstellungsfehler durch Sikuli-basierte Tests aufgedeckt. Außerdem wird durch die Trennung von Testsoftware und zu testender Software sichergestellt, dass keine Integrations- und Nebenläufigkeitsprobleme durch die Einbettung von fremdem Code in das zu testende Projekt entstehen können.
Diese Vorteile werden durch einen höheren Pflegeaufwand für die Tests erkauft, der gerade bei langfristig angelegten Projekten nicht zu vernachlässigen ist. So müssen die Tests an die sich verändernden User-Interfaces neuer Betriebssysteme angepasst werden. Ebenso müssen Veränderungen an dem User-Interface des zu testenden Projektes gepflegt werden. Auch sind die Tests durch das Screen-Capturing stärker hardwaregebunden als herkömmliche Testmethoden. Für das TextGrid-Projekt hat der höhere Nutzen diesen Aufwand jedoch gerechtfertigt.
Tests werden mit Sikuli in einem Python-Dialekt entwickelt. Die Integration einer vollwertigen Programmiersprache in das Testsystem gestattet es, dynamisch in laufende Tests einzugreifen und Testparameter abhängig von Ereignissen zu verändern.

Abbildung 1 Quellcode aus Sikuli

[1] http://www.sikuli.org/
[2] http://www.eclipse.org/swtbot/

2. Aufbau des Testsystems

Die Aufgabe der Hochschule Worms bei den funktionalen Tests war es, möglichst real das parallele Arbeiten unterschiedlicher Nutzer am TextGrid-System zu simulieren. Hierzu wurden die Testszenarien in kleine, abgeschlossene Use-Cases unterteilt. Da die Use-Cases abgeschlossen waren, war der Zustand des Systems nach Abschluss jedes Tests identisch. Dies ermöglichte es, die Tests auf den verschiedenen Testclients in unterschiedlicher Reihenfolge auszuführen, sodass mehrere Clients eine Aufgabe bearbeiten konnten, während andere Clients komplett anderen Tätigkeiten nachgingen.
Da TextGrid ein Multiuser-System ist, war es von entscheidender Bedeutung, dies in den automatisierten Tests entsprechend zu berücksichtigen. Das Zentrum für Technologietransfer und Telekommunikation (ZTT) der Hochschule Worms hat deshalb einen zentralen Webservice entwickelt, welcher einen Pool von Testusern verwaltet und während der Testläufe jedem Testclient eigene Login-Credentials zuweist. Dieses System konnte flexibel konfiguriert werden, so dass es auch möglich war, zu simulieren, dass mehrere User mit den gleichen User-Credentials arbeiteten.
Da nun ein zentraler Server vorhanden war, wurde dieser ausgebaut, um die Testszenarien noch exakter bestimmen zu können. Dies ermöglichte es zum einen, einzelne Use-Cases von den Tests auszuschließen und zum anderen, bestimmte Parameter wie Arbeitsdauer eines Use-Cases oder für den Test zu verwendende Dateien für jeden Test-Client separat festzulegen.

3. Praktische Durchführung der funktionalen Tests

Die funktionalen Tests wurden in der Regel in einem Labor der Hochschule Worms mit 21 Arbeitsplätzen durchgeführt. Ein solcher Umfang bringt mehrere logistische Herausforderungen mit sich. Als erstes muss sichergestellt sein, dass alle Clients den Test parallel starten, ohne dass durch den Testaufbau selbst Engstellen verursacht werden, beispielsweise im lokalen Netzwerk.
Anschließend ist wichtig, dass alle Clients die gleiche Version der funktionalen Tests ausführen, da sonst gegebenenfalls Fehler gemeldet werden, die nicht oder nicht mehr existieren. Die Fehlermeldungen selbst sind mitunter nicht aussagekräftig. Häufig ist es notwendig, zu sehen, was vor dem Auftreten des Fehlers geschehen ist. Diese Problematik konnte ebenfalls gelöst werden. Die Tests selbst werden über ein komplexes Programm gestartet, welches direkt nach dem Systemstart ausgeführt wird. Dieses Programm startet den Test der einzelnen Clients versetzt über einen Zeitraum von 60 Sekunden. Bei einer durchschnittlichen Laufzeit einer Testsitzung von 45 Minuten hat der verzögerte Start keine Auswirkungen auf die Ergebnisse. Zuvor wird die neueste Version der Testsoftware aus einem zentralen Sourcecode Repository geladen und auf den jeweiligen Clients installiert. Ebenso wird die neueste Version des TextGridLab von einem zentralen Server heruntergeladen und installiert. Direkt vor dem Start des Tests wird mittels der Software ScreenFlow[3] eine Videoaufzeichnung des gesamten Tests begonnen und erst nach Abschluss des Tests wieder beendet. Wenn der Test fehlerfrei ablief, wurde die Video-Aufzeichnung zusammen mit allen Logdateien gelöscht. Wenn jedoch ein Fehler auftrat, wurde die Videodatei zusammen mit den Logdateien auf einen zentralen Server kopiert, auf dem die Daten zuerst überprüft und später allen TextGrid-Entwicklern zur Verfügung gestellt werden konnten.

[3] http://www.telestream.net/screenflow/

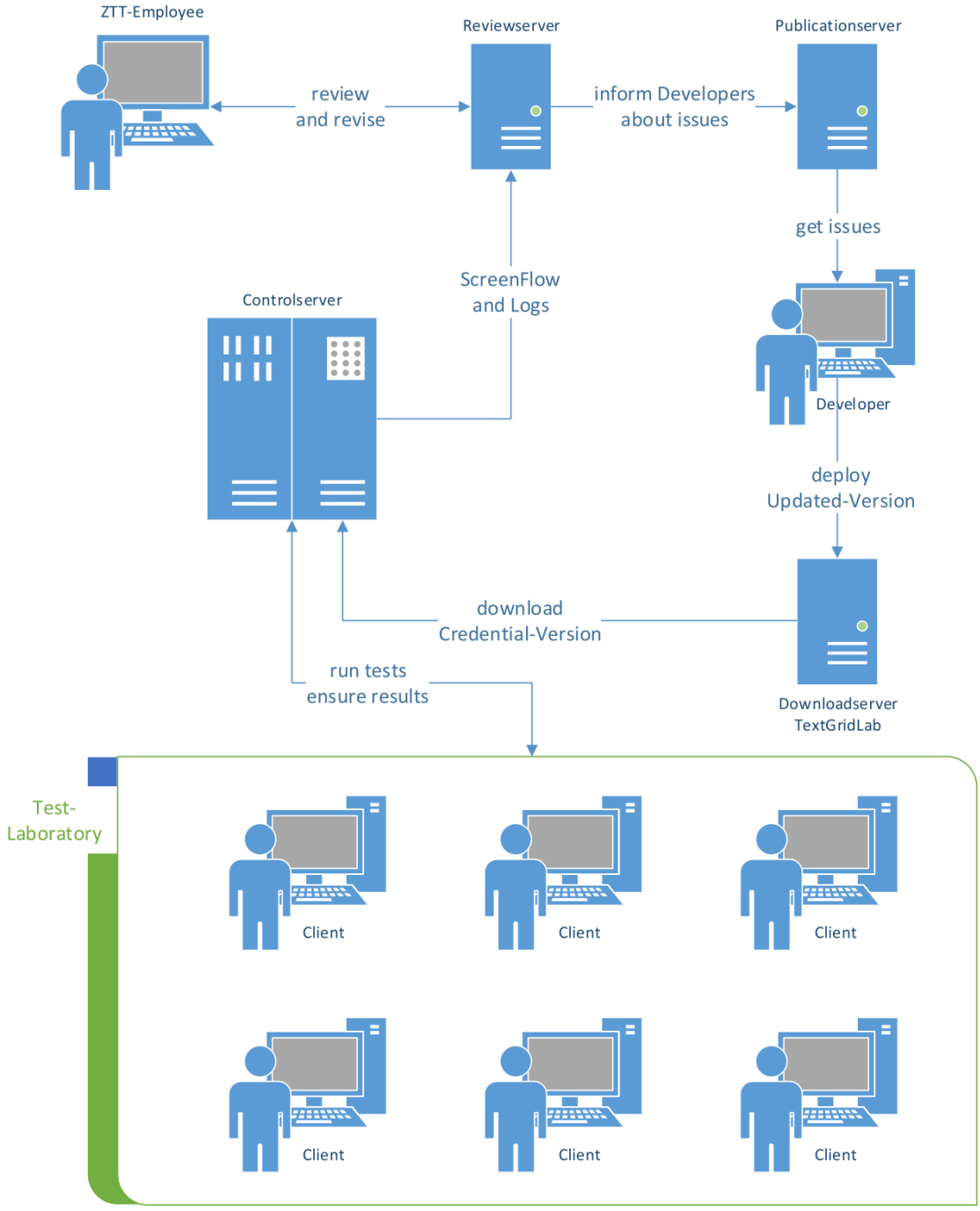
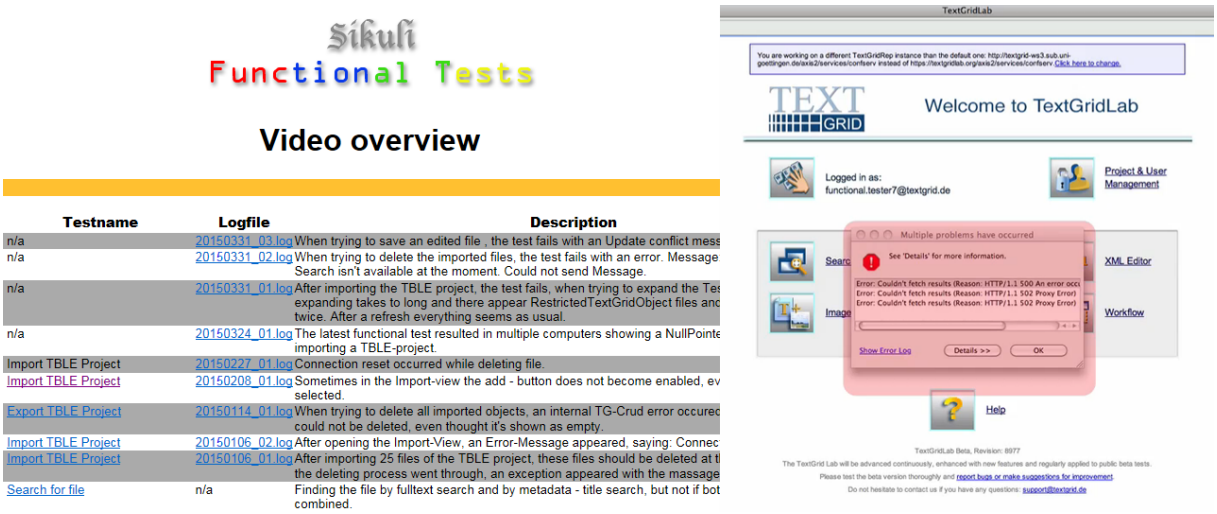


Abbildung 2 Schematische Darstellung der funktionalen Testumgebung von TextGrid

Zu Beginn der Phase 3 des TextGrid-Projektes existierten rudimentäre Testszenarien für die häufig benötigten Funktionen „Anmelden am TextGrid-Repository“, „Erstellen neuer Projekte“, „Erstellen neuer Textdateien“, „Im- und Export von Texten und Bildern“ und „Editieren einer XML-basierten Textdatei über einen längeren Zeitraum“. Diese Tests standen für die Betriebssysteme Windows XP und OS X 10.6 zur Verfügung.
Im Rahmen der Projektphase 3 von TextGrid hat das ZTT diese Tests auf die Betriebssystemversionen Windows 7 und OS X 10.7, 10.8 und 10.9 portiert. Eine Portierung der Tests auf Linux-Betriebssysteme konnte wegen der Vielzahl der Kombinationsmöglichkeiten aus Linux-Distribution und Windowmanager nicht durchgeführt werden. Ergebnisse wären nur für jeweils eine Kombination aus Distribution und Windowmanager repräsentativ gewesen. Weiterhin wurden Änderungen an den User-Interfaces aller TextGrid-Views und –Perspectives in die Tests übernommen. Im Rahmen der Weiterentwicklung der Tests setzte das ZTT neue Testszenarien nach Vorgaben der Fachwissenschaftler um und konnte so häufig benötigte Funktionalitäten simulieren. Hierzu zählten unter anderem Tests zum Copy-Workflow, dem Vervollständigen der Nutzerdaten, dem Metadaten-Editor und dem Text-Text-Link-Editor(TTLE).
Vor jedem offiziellen Release einer neuen TextGrid-Version (hierzu zählten auch Minor Update-Versionen) wurden iterativ Tests in den Laboren der Hochschule durchgeführt, bei denen möglichst viele parallel arbeitende Clients simuliert wurden. Bereits während der Durchführung der Tests wurden die TextGrid-Entwickler über auftretende Probleme informiert, sodass Fehler schnellst-möglich behoben werden konnten. Dieser agile Ansatz ermöglichte es, mehrere Testläufe pro Tag durchzuführen, um so die Testphase vor dem Release möglichst kurz halten zu können.


Sikuli
Functional Tests

Video overview

Testname	Logfile	Description
n/a	20150331_03.log	When trying to save an edited file , the test fails with an Update conflict mess
n/a	20150331_02.log	When trying to delete the imported files, the test fails with an error. Message Search isn't available at the moment. Could not send Message.
n/a	20150331_01.log	After importing the TBLE project, the test fails, when trying to expand the Tes expanding takes to long and there appear RestrictedTextGridObject files and twice. After a refresh everything seems as usual.
n/a	20150324_01.log	The latest functional test resulted in multiple computers showing a NullPointe importing a TBLE-project.
Import TBLE Project	20150227_01.log	Connection reset occurred while deleting file.
Import TBLE Project	20150208_01.log	Sometimes in the Import-view the add - button does not become enabled, ev selected.
Export TBLE Project	20150114_01.log	When trying to delete all imported objects, an internal TG-Crud error occurec could not be deleted, even thought it's shown as empty.
Import TBLE Project	20150106_02.log	After opening the Import-View, an Error-Message appeared, saying: Connec
Import TBLE Project	20150106_01.log	After importing 25 files of the TBLE project, these files should be deleted at t the deleting process went through, an exception appeared with the massage
Search for file	n/a	Finding the file by fulltext search and by metadata - title search, but not if bot combined.

Abbildung 3 Website der funktionalen Testergebnisse und entsprechenden Videos mit Fehlermeldungen

4. Erweiterung des Testsystems

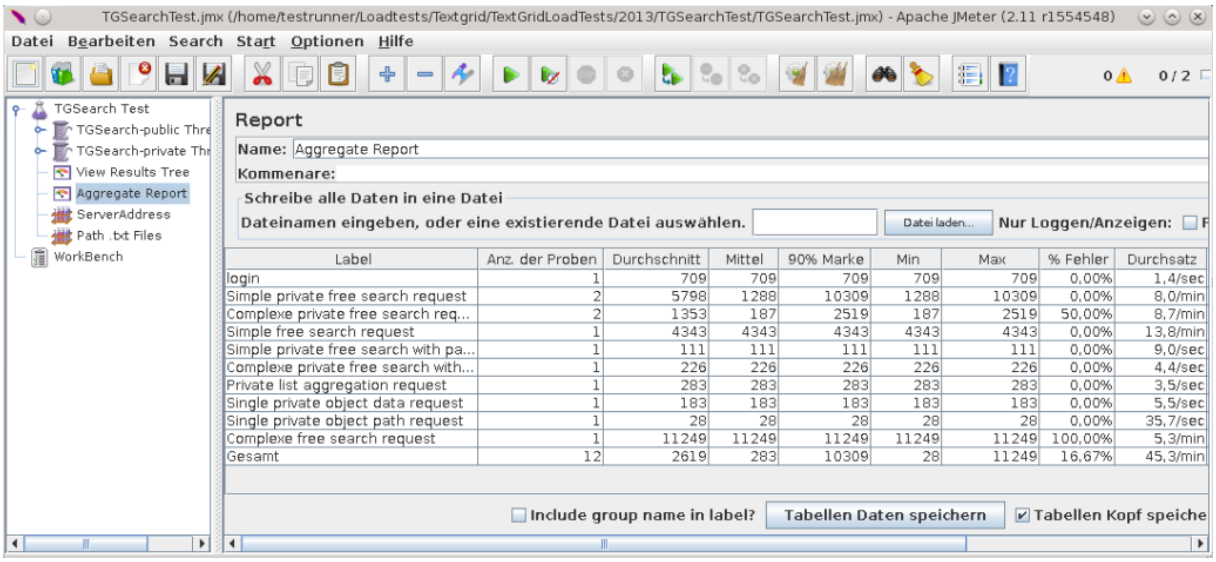
4.1. Integration der funktionalen Tests in den Buildprozess

Auf Wunsch des Konsortiums wurde weiterhin die Integration der funktionalen Tests in den automatisierten Buildprozess von TextGrid evaluiert. Eine automatisierte Durchführung der Tests, direkt nach der Bereitstellung geänderten Quellcodes, gibt Entwicklern kurzfristig Feedback darüber, ob Änderungen am Quellcode Auswirkungen in anderen Programmteilen verursachen.
Das TextGrid-Buildsystem basiert auf einem Jenkins-Server. Um die Integration der funktionalen Tests in den Buildprozess prototypisch zu entwickeln, war es zuerst erforderlich, einen entsprechenden Test-Server aufzubauen. Eine Verbindung vom Buildserver zum Testprozess herzustellen war problemlos möglich. Eine automatisierte Durchführung der Tests war in diesem Aufbau jedoch nicht möglich, da das entsprechende Hochschul-Labor nicht 24 Stunden pro Tag für Tests zur Verfügung stand. Es wurde deshalb versucht, die Testclients mit Virtualisierungslösungen auf eine serverbasierte Grundlage zu stellen. Da ein über Screen-Capturing arbeitender Testaufbau in gewissen Maßen von den Bildschirmeinstellungen abhängt, virtualisierte Systeme sich jedoch Bildschirmeinstellungen teilen, war es nicht möglich, mit virtualisierten Testsystemen zufriedenstellende Ergebnisse zu erzielen. Der Einsatz dedizierter Serverhardware ermöglichte letztendlich zwar eine erfolgreiche Integration der funktionalen Tests in den Buildprozess, entsprechende Hardware stand jedoch nicht in ausreichender Menge zur Verfügung, um aussagekräftige Testergebnisse zu erhalten.

4.2. Integration der Lasttests in die funktionalen Tests

Um die Qualität der Testergebnisse weiter zu verbessern, wurde die Durchführung der funktionalen Tests mit auf der Software jMeter[4] basierenden Lasttests gekoppelt. Das ZTT hatte bereits in der Projektphase 2 mit der Entwicklung von Lasttests begonnen; damals um die Stabilität der kritischen Komponente eXist zu gewährleisten. Zur Unterstützung der funktionalen Tests wurden vor jedem Testlauf Lasttests gegen die Backend-Komponenten von TextGrid durchgeführt. Dies war notwendig, da insbesondere netzwerkbasierte Fehler nicht immer als solche auf dem Bildschirm oder in den Logdateien wiedergegebenen werden. Da Netzwerkzugriffe häufig im Hintergrund geschehen, ist es bei der Fehleranalyse oft nicht ersichtlich, dass undefiniertes Verhalten des Test-Programms aus einem Netzwerkproblem resultiert. Entsprechend diesen Anforderungen wurden in TextGrid Phase 3 Lasttests für die Komponenten TG-CRUD, TG-SEARCH und TG-AUTH entwickelt und über die Projektlaufzeit hinweg an Änderungen der Remote-Interfaces der jeweiligen Dienste angepasst.
Da die funktionalen Tests nur nach erfolgreich absolvierter Durchführung der Lasttests ausgeführt wurden, konnte sichergestellt werden, dass alle, bei der Ausführung der funktionalen Tests gefundenen Fehler ursächlich im TextGridLab beheimatet waren.

[4] http://jmeter.apache.org/



Label	Anz. der Proben	Durchschnitt	Mittel	90% Marke	Min	Max	% Fehler	Durchsatz
login	1	709	709	709	709	709	0,00%	1,4/sec
Simple private free search request	2	5798	1288	10309	1288	10309	0,00%	8,0/min
Complexe private free search req...	2	1353	187	2519	187	2519	50,00%	8,7/min
Simple free search request	1	4343	4343	4343	4343	4343	0,00%	13,8/min
Simple private free search with pa...	1	111	111	111	111	111	0,00%	9,0/sec
Complexe private free search with...	1	226	226	226	226	226	0,00%	4,4/sec
Private list aggregation request	1	283	283	283	283	283	0,00%	3,5/sec
Single private object data request	1	183	183	183	183	183	0,00%	5,5/sec
Single private object path request	1	28	28	28	28	28	0,00%	35,7/sec
Complexe free search request	1	11249	11249	11249	11249	11249	100,00%	5,3/min
Gesamt	12	2619	283	10309	28	11249	16,67%	45,3/min

Abbildung 4 Lasttestergebnisse mit JMeter

5. Ergebnisse

Durch dieses ausführliche Testkonzept, konnte während der gesamten dritten Projektphase von TextGrid sichergestellt werden, dass alle Releases ohne das Auftreten von Problemen durchgeführt werden konnten, was zu einer positiven Erfahrung für die Nutzer und damit zu einer verbesserten Annahme des Projektes geführt hat. Aufgrund der positiven Erfahrungen mit den Sikuli-basierten funktionalen Tests im Projekt TextGrid 3 wurde außerdem beschlossen, dieses Test-Verfahren für das Projekt XML-Print zu übernehmen.