



# Scone

# UserTestTool –

## Kurzeinführung und Referenz

**Torsten Haß**

Diplomarbeit in den  
Arbeitsbereichen ASI und VSIS  
Fachbereich Informatik  
Universität Hamburg

Das UserTestTool wurde entwickelt, um partizipative Gebrauchstauglichkeitstests von Websites zu erleichtern. Dieses Werkzeug unterstützt den Tester, indem es ihm viele nötige Routineaufgaben abnimmt und dadurch den Zeitbedarf dieser Testmethode verringert. Das UserTestTool zeigt der Testperson die zu erfüllenden Aufgaben und speichert die Eingaben der Testperson. Des Weiteren kann das UserTestTool den Internet Explorer steuern und fast alle Aktionen der Testperson, wie angeklickte Links, besuchte URLs und vieles mehr speichern.

Das UserTestTool ist im Rahmen einer Diplomarbeit an der Universität Hamburg, Fachbereich Informatik, Arbeitsbereiche ASI und VSIS entwickelt worden. Fragen, Anregungen und Kritik senden Sie bitte an [Torsten.Hass@hansenet.de](mailto:Torsten.Hass@hansenet.de).

# Inhaltsverzeichnis

1	Einleitung .....	3
2	Funktionsweise.....	3
2.1	Definition eines Tests.....	3
3	Download .....	4
4	Installation.....	4
5	Konfiguration .....	4
6	Scone starten und beenden .....	7
7	Definition der Testbeschreibungdatei.....	8
8	Protokolldatei .....	9
9	Fragen, Anregungen und Kritik .....	10
A	Anhang Komponentenreferenz .....	11
B	Anhang Einträge der Protokolldatei.....	20

# 1 Einleitung

Tests, in denen Websites mit „echten“ Benutzern getestet werden, sind sehr zeit- und arbeitsaufwendig: Die Äußerungen der Testperson müssen analysiert, ihre Aktionen, ihre Eingaben und die besuchten Internetseiten protokolliert und später ausgewertet werden. Zusätzlich müssen die Aufgaben erklärt werden, ohne die Testperson zu beeinflussen.

Das UserTestTool wurde entwickelt, um den Tester zu unterstützen. Es nimmt dem Tester viele Routineaufgaben ab und verringert dadurch den Zeitbedarf dieser Testmethode. Das UserTestTool zeigt der Testperson die zu erfüllenden Aufgaben und speichert die Eingaben der Testperson. Des Weiteren kann das UserTestTool den Internet Explorer steuern und fast alle Aktionen der Testperson, wie angeklickte Links, besuchte URLs und vieles mehr speichern.

## 2 Funktionsweise

Ein kompletter Test besteht aus mehreren Aufgaben, z. B. das Suchen von Informationen auf Websites, das Beantworten von Fragen. Jede Aufgabe besteht aus Texten, Schaltflächen oder Eingabemöglichkeiten, die der Testperson in einem Fenster präsentiert werden. Der Inhalt dieser Aufgabenfenster wird in einer XML-Datei definiert, in der die anzuzeigenden Texte, Schaltflächen und Eingabemöglichkeiten aufgeführt werden, und deren Verhalten festgelegt wird. Die Schaltflächen können bei Betätigung bestimmte Aktionen ausführen, z. B. andere Komponenten aktivieren, deaktivieren, hervorheben, Stoppuhren starten oder auslesen, Internetseiten im Browser öffnen oder die nächste Aufgabe starten. Für die Testperson steht ein Textfeld für die Texteingabe und eine Auswahlliste zum Auswählen aus vordefinierten Texten zur Verfügung. Alternativ kann eine Likert-Skala eingesetzt werden, die die Auswahl der Testperson als numerischen Wert speichert.

Die von der Testperson eingegebenen Daten, die angeklickten Schaltflächen und viele Aktionen am Webbrowser werden gespeichert. Die gesammelten Daten werden im XML- und im Textformat abgelegt und können in andere Programme, z. B. Microsoft Excel, importiert werden. Eine genaue Beschreibung aller Einträge der Protokolldateien finden Sie im Anhang A.

### 2.1 Definition eines Tests

Der Ablauf eines solchen Tests mit all seinen Aufgaben wird in einer XML-Datei festgelegt. Jede Aufgabe wird in einem eigenen Bereich der XML-Datei definiert. Dabei wird die Aufgabe aus festgelegten Komponenten wie Text-, Schaltflächen- und Texteingabefeld-Komponenten zusammengesetzt. Dem Programmpaket liegt eine Beispieldatei

bei, die den Dateiaufbau demonstriert und sich leicht um Komponenten oder Aufgaben erweitern lässt. Häufig wiederkehrende Komponenten oder Komponentengruppen lassen sich im Layout-Bereich der Datei vordefinieren. Diese können dann eingebunden und für die aktuelle Situation modifiziert werden.

### 3 Download

Das UserTestTool läuft innerhalb des Frameworks Scone ab. Scone ist ein Java-Framework, das die Programmierung und Evaluation von Navigationswerkzeugen unterstützt. Eine vorkonfigurierte Version von Scone, in der das UserTestTool enthalten ist, liegt unter

<http://www.scone.de/userTestTool.html>

zum Download bereit.

Das UserTestTool läuft unter Windows NT, 2000 und XP. Als Browser wird Microsofts Internet Explorer in der Version 5.0, 5.5 oder 6.0 benötigt.

Des Weiteren sind folgende Komponenten nötig:

- Java 2 Platform, Standard Edition (J2SE) Version 1.4.0 oder höher. Download unter: <http://java.sun.com/j2se/downloads.html>

### 4 Installation

- Installieren Sie das Java Development Kit (JDK), sofern es nicht bereits installiert ist.
- Installieren Sie Scone, indem Sie es in ein Verzeichnis Ihrer Wahl entpacken. Bitte benutzen Sie nur die vorkonfigurierte Version von der oben angegebenen Internetseite. Darin ist das UserTestTool bereits enthalten und vorkonfiguriert.

### 5 Konfiguration

In Microsofts Internet Explorer müssen einige Einstellungen Vorgenommen werden, damit er über Scone auf das Internet zugreift. Scone muss als Proxy-Server eingerichtet werden, „\*.scone.de“ muss als vertrauenswürdige Site bekannt gemacht werden, das Cache muss ausgeschaltet und die Kommunikation mit Applets erlaubt werden. Außerdem muss Windows dem Internet Explorer erlauben, das Browserfenster zum obersten Fenster zu machen.

Im Verzeichnis „scone/setup“ befindet sich eine Datei mit dem Namen „Config\_IE\_5+6\_For\_Scone.reg“. Starten Sie diese, um alle erwähnten Einstellungen vorzunehmen. Um die Einstellungen rückgängig zu machen, starten sie die Datei mit

dem Namen „Config\_IE\_5+6\_No\_Scone.reg“, die sich im gleichen Verzeichnis befindet. Die Einstellungen werden erst nach einem Neustart übernommen.

Die Proxy-Einstellungen dieser Konfigurationsdateien beziehen sich nur auf Verbindungen über die Netzwerkkarte. Im Fall von Internetverbindungen über Modem oder ISDN muss der Proxy-Server für die entsprechende DFÜ-Verbindung von Hand eingestellt werden. Das wird im nächsten Absatz beschrieben.

Wenn Sie über die Netzwerkkarte ins Internet gehen und, wie oben beschrieben, die Datei „Config\_IE\_5+6\_For\_Scone.reg“ ausgeführt haben, können Sie nun zum nächsten Kapitel springen.

Alternativ gibt es die Möglichkeit, die Einstellungen von Hand vorzunehmen. Die Proxy-Server-Einstellungen befinden sich im Internet Explorer unter dem Menü *Extras / Internetoptionen...* und dort auf der Registerkarte *Verbindungen*. Zum Einstellen des Proxy-Servers für Internetverbindungen über die Netzwerkkarte klicken Sie bitte im Bereich *LAN-Einstellungen* auf die Schaltfläche *Einstellungen* (siehe Abbildung 1). Zum Einstellen des DFÜ-Proxy-Servers klicken Sie bitte auf die Schaltfläche *Einstellungen* im Bereich *DFÜ- und VPN-Einstellungen*.

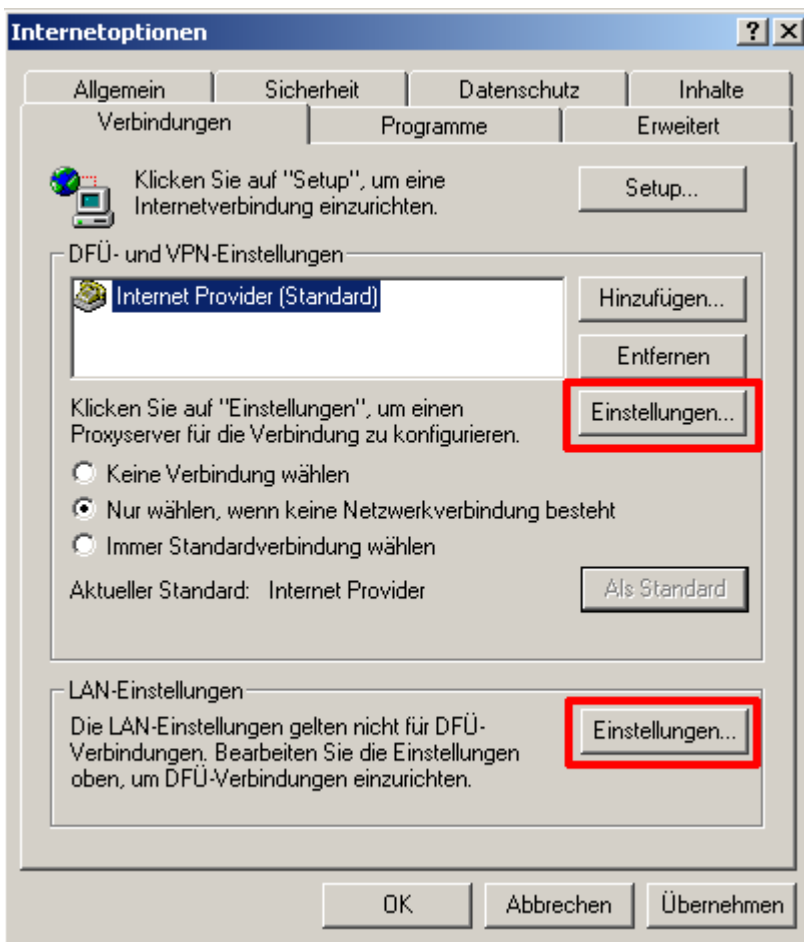


Abbildung 1: Internetoptionen des Internet Explorers

Im erscheinenden Fenster aktivieren Sie das Kästchen vor *Proxyserver für...* und geben als Adresse: *localhost*, und als Port *8088* an (siehe Abbildung 2 für Internetverbindungen über die Netzwerkkarte und Abbildung 3 für Internetverbindungen über Modem oder ISDN).

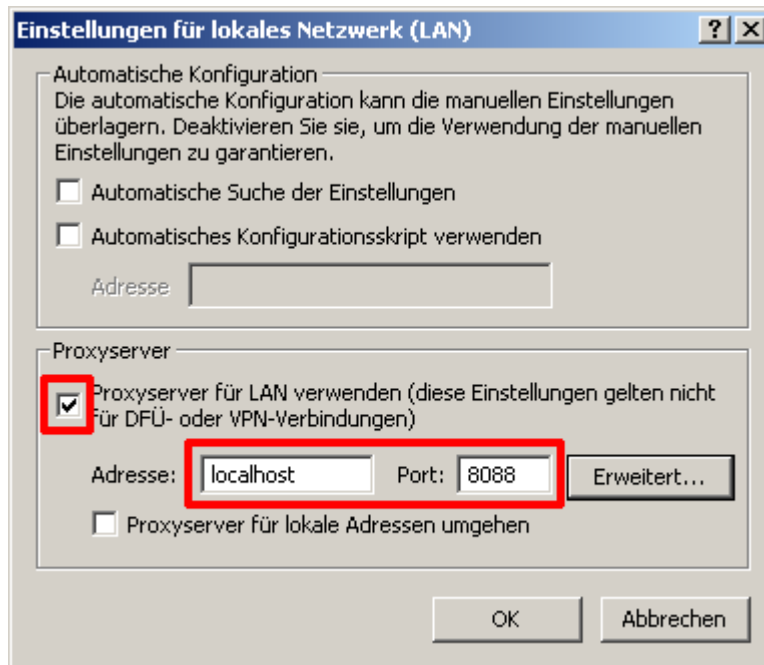


Abbildung 2: Einstellungen für Verbindungen über das lokale Netzwerk

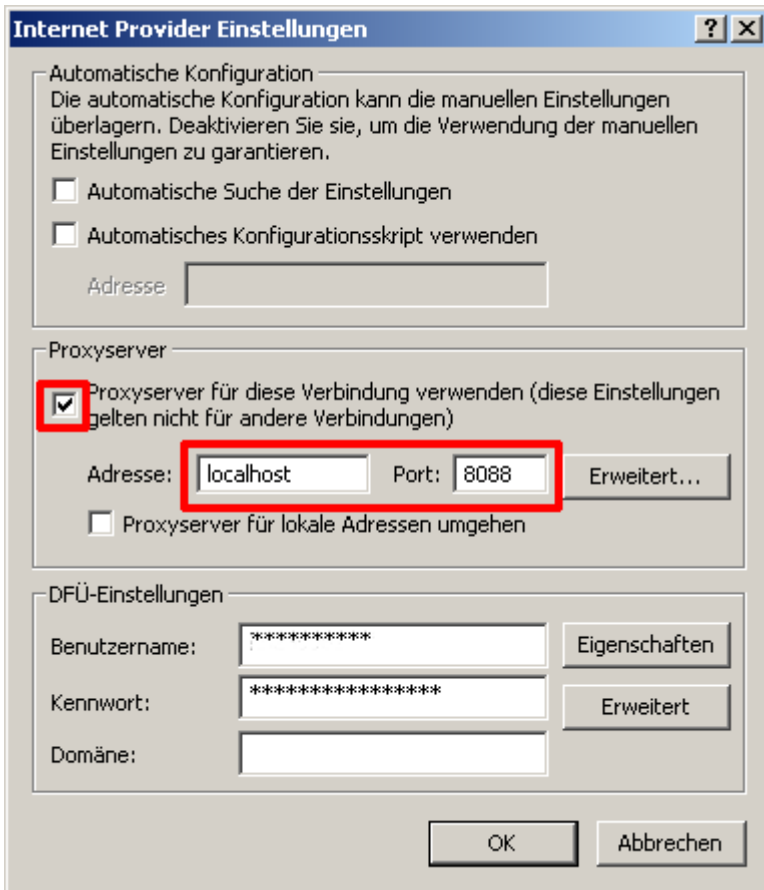


Abbildung 3: Einstellungen für Internetprovider über Modem oder ISDN

Zum Deaktivieren des Cache öffnen Sie bitte das Fenster *Internetoptionen* über das Menü *Extras / Internetoptionen...* und klicken Sie auf die Registerkarte *Allgemein*. Im Bereich *Temporäre Internetdateien* klicken Sie auf die Schaltfläche *Einstellungen*. In dem Fenster *Einstellungen* aktivieren Sie den Punkt *Bei jedem Zugriff auf die Seite*.

Die übrigen Einstellungen der Reg-Datei sind für das UserTestTool nicht erforderlich.

## 6 Scone starten und beenden

Um Scone zu starten, rufen Sie bitte die Datei „runScone.bat“ in Scones run-Verzeichnis auf. Sobald Scone in seinem Textfenster die Meldung **Console>** ausgibt, können Sie das UserTestTool starten. Dazu öffnen Sie den Internet Explorer und rufen Sie die Seite „usertest.scone.de“ auf. Scone wird daraufhin auf der linken Bildschirmhälfte ein Auswahlfenster erzeugen. Wählen Sie unter Testbeschreibung den Beispieltest namens **Beispiel** aus, geben Sie einen Teilnehmernamen ein und klicken Sie auf „Neuen Test starten“. Dieses Beispiel sollte sich selbst erklären.

Sollten während des Ablaufes Fehler auftreten, z. B. bei fehlerhaften Testbeschreibungsdateien, dann werden diese in dem Textfenster ausgegeben, in dem Scone

gestartet wurde. Während des Starts werden alle Testbeschreibungsdateien überprüft und gefundene Fehler der XML-Struktur ausgegeben.

Um Scone zu beenden wechseln Sie zu dem Textfenster, in dem Scone gestartet wurde. Geben Sie dort den Befehl **quit** ein, um das UserTestTool und Scone zu beenden.

## 7 Definition der Testbeschreibungsdatei

In diesem Abschnitt wird der grobe Aufbau der XML-Testbeschreibungsdatei und der XML-Knoten beschrieben, aus denen die XML-Testbeschreibungsdatei zusammengesetzt werden kann. Die Testbeschreibungsdateien liegen in dem Verzeichnis „**scone/run/config/userTestDescriptions**“. Dort finden Sie auch die **Beispiel.xml**-Datei, die Sie gern als Vorlage benutzen können. Eine detaillierte Beschreibung der möglichen XML-Knoten finden Sie in der Komponentenreferenz im Anhang A.

Wichtig für eine funktionierende Testbeschreibungsdatei ist, dass sie auf jeden Fall den „**?xml**“-Knoten (Abschnitt A.1), mindestens einen „**task**“-Knoten (Abschnitt A.6) und als dessen Kindknoten die gewünschten Komponentenknoten enthält.

Jeder „**task**“-Bereich enthält genau eine Aufgabe. Im Bereich „**layout**“ gibt es mehrere „**template**“-Bereiche, in denen Komponenten vordefiniert werden. Diese Templates werden in die einzelnen Aufgaben eingebunden.

Des Weiteren ist zu beachten, dass XML die Groß-Kleinschreibung beachtet und keine ungeschlossenen Knoten erlaubt. Das heißt: jeder Knoten (z. B. **<text ...>**) muss durch einen abschließendes Tag (z. B. **</text>**) geschlossen werden. Alternativ kann ein Knoten im öffnenden Tag gleich geschlossen werden (z. B. **<text ... />**). Es ist auch nicht erlaubt, Attribute ohne Gleichheitszeichen und folgenden Wert in Anführungszeichen zu notieren. Attribute müssen immer in der Form **attributName="wert"** innerhalb eines Knotens angegeben werden. Weitere Informationen zu XML finden Sie im Praktikum Internet des Fachbereichs Informatik unter:

[http://print-www.informatik.uni-hamburg.de/Dokumentation/09\\_XML.pdf](http://print-www.informatik.uni-hamburg.de/Dokumentation/09_XML.pdf)

oder auf den Internetseiten von Sun unter

<http://java.sun.com/webservices/docs/1.0/tutorial/doc/IntroXML.html>

Jede Komponente hat einen Namen, der ihr über das Attribut „**name**“ mitgegeben wird. Diese Namen dienen dazu, Komponenten während der Laufzeit zu beeinflussen. Diese Namen sind nur lokal für die aktuelle Aufgabe gültig. Komponenten anderer Aufgaben lassen sich nicht beeinflussen. Lediglich die Namen der Stoppuhrkomponenten: „**stopWatchStart**“, „**stopWatchElapsed**“ und „**stopWatchLapTime**“ sind

aufgabenübergreifend gültig, um Zeitmessungen über mehrere Aufgaben und „Zeitspanne pro Aufgabe“ zu ermöglichen.

Jede Komponente hält 10 Pixel Abstand zur folgenden. Dieser Wert kann verändert werden, indem die obere Komponente das Attribut **bottomPadding** benutzt. Auf diese Weise lassen sich Komponenten räumlich von einander trennen oder zusammenziehen.

## 8 Protokolldatei

Alle Eingaben, Maus- und Tastaturaktionen der Testperson werden in Protokolldateien gespeichert. Für jeden Test werden zwei Protokolldateien erzeugt: eine im XML-Format und eine im TDF-Format (Tab Delimited File).

Diese Dateien werden im Verzeichnis „**scone/run/log/userTestResults**“ abgelegt. Der Dateiname der jeweiligen Protokolldatei setzt sich aus dem Namen der Testbeschreibungdatei, einer vierstelligen laufenden Nummer und dem Namen der Testperson zusammen. Eine typische Protokolldatei hätte beispielsweise den Namen **BeispielTest.0002.Meier.xml**. Genau wie die Testbeschreibungdatei ist auch die XML-Protokolldatei in Aufgaben („task“-Knoten) unterteilt. Innerhalb einer Aufgabe werden zunächst alle Aktionen der Testperson abgelegt, danach sind die Daten der Textfelder und andere Eingabekomponenten gespeichert.

Jeder Eintrag innerhalb einer Aufgabe hat einen Knotennamen, der die Herkunft der Daten angibt. Die Attribute der Knoten geben die übrigen Daten an. So hat jeder Knoten innerhalb einer Aufgabe ein „**action**“-Attribut, in dem die Eingabe oder Aktion der Testperson gespeichert ist. Das „**name**“-Attribut gibt den Namen der Komponente an, von dem die Daten stammen. Jeder Komponente wurde in der Testbeschreibungdatei ein Name zugewiesen. Das „**timestamp**“-Attribut speichert den Zeitpunkt der Aktion, z. B. bei Schaltflächen, oder den Zeitpunkt des Auslesens, z. B. bei Textfeldern.

Eine genaue Liste der möglichen Einträge finden Sie im Anhang B: Einträge der Protokolldatei.

Die Protokolldatei im TDF-Format enthält die gleichen Einträge wie die XML-Protokolldatei. Jede Zeile beginnt mit dem Knotennamen. Es folgt der Komponenten- oder Framename, danach die Aktionsdaten, die Zeitangabe und der URI.

Die einzelnen Werte sind durch Tabstops voneinander getrennt. Auf diese Weise lässt sich die Datei in Programme wie Microsoft Excel importieren (siehe Abbildung 4).

	A	B	C	D	E	F	G	H
1	userTestResult	Torsten	config/userTestDes	Mon Nov 24 14:53:49	03. Mrz			
2	task	Einleitung		1,06968E+12				
3	buttonClick	nextButton	click	1,06968E+12				
4	task	Aufgabe 1		1,06968E+12				
5	buttonClick	Start	click	1,06968E+12				
6	browserControl	SCONE106968	browserToFront	1,06968E+12				
7	browserControl	SCONE106968	openUri	1,06968E+12	http://www.informatik.uni-hamburg.de			
8	browser	SCONE106968	newUrl	1,06968E+12	http://www.informatik.uni-hamburg.de/			
9	link	SCONE106968	click	1,06968E+12	http://www.informatik.uni-hamburg.de/bib/			
10	link	SCONE106968	click	1,06968E+12	http://www.informatik.uni-hamburg.de/bib/allgemein			
11	buttonClick	Stop	click	1,06968E+12				
12	browserControl	SCONE106968	openUri	1,06968E+12	http://usertest.scone.de/blank			
13	stopWatch	SearchTime	elapsedTime		51854			
14	buttonClick	Next	click	1,06968E+12				
15	textFieldText	Zeiten	Mo-Fr, 9-19 Uhr	1,06968E+12				
16	task	Fragebogen		1,06968E+12				
17	buttonClick	endButton	click	1,06968E+12				
18	dropDownBoxText	LogoDesign	Geht so	1,06968E+12				
19	dropDownBoxValue	LogoDesign	10	1,06968E+12				
20	likertScaleSelection	performanz	4	1,06968E+12				
21	likertScaleSelection	Kompliziert	6	1,06968E+12				
22	likertScaleSelection	Unterteilungen	3	1,06968E+12				

Abbildung 4: Microsoft Excel mit der importierten Protokolldatei

## 9 Fragen, Anregungen und Kritik

Das UserTestTool ist im Rahmen einer Diplomarbeit am Fachbereich Informatik der Universität Hamburg entstanden. Fragen, Anregungen und Kritik senden Sie bitte an Torsten Haß, E-Mail: [Torsten.Hass@hansenet.de](mailto:Torsten.Hass@hansenet.de). Anregungen und konstruktive Kritik werden sowohl in das Werkzeug, als auch in die Diplomarbeit eingebaut.

## A Anhang Komponentenreferenz

In diesem Abschnitt werden die XML-Knoten beschrieben, aus denen die XML-Testbeschreibungsddatei zusammengesetzt werden kann. Die Testbeschreibungsddateien liegen in dem Verzeichnis „`scone/run/config/userTestDescriptions`“.

### A.1 „?xml“

Der „?xml“-Knoten informiert den XML-Parser über die verwendete XML-Version und die Zeichenkodierung. Dieser muss sollte in der ersten Zeile der Testbeschreibungsddatei stehen und die folgende Form haben:

```
<?xml version='1.0' encoding='ISO-8859-1'?>
```

Diese Zeichenkodierung sollte der UTF-8-Kodierung vorgezogen werden, da viele Editoren diese Kodierung verwenden und damit die deutschen Umlaute in der Testbeschreibungsddatei korrekt erkannt werden.

### A.2 „button“

Der „button“-Knoten erzeugt eine Schaltfläche in dem Aufgabenfenster. Ein Beispiel für eine Schaltflächendefinition ist die folgende:

```
<button name="Start" text="Aufgabe starten" enabled="false"
highlighted="true" bottomPadding="30">
  <browserToFront/>
  <browserResize width="800" height="600"/>
  <openUri>www.informatik.uni-hamburg.de</openUri>
  <enable>Stop</enable>
  <disable>Start</disable>
  <highlight>Stop</highlight>
  <stopWatchStart>SearchTime</stopWatchStart>
</button>
```

Folgende Attribute sind möglich:

- „**name**“ gibt dem Button einen Namen. Mit Hilfe dieses Namens kann die Schaltfläche durch andere Schaltflächen aktiviert werden und die Aktion der Schaltfläche in der Protokolldatei identifiziert werden.
- „**text**“ enthält die Beschriftung der Schaltfläche.
- „**enabled**“ legt fest, ob die Schaltfläche aktiviert und benutzbar („**true**“) oder ausgegraut und unbenutzbar („**false**“) ist. Wird das Attribut „**enabled**“ nicht angegeben, ist die Schaltfläche automatisch aktiviert.
- „**highlighted**“ gibt an, ob die Schaltfläche gelblich hinterlegt und damit hervorgehoben dargestellt werden soll.
- „**bottomPadding**“ legt die Anzahl der Pixel fest, die diese Komponente räumlich von der nächsten trennen.

Die Kindknoten des „**button**“-Knotens stehen für Aktionen, die beim Anklicken der Schaltfläche ausgeführt werden. Es werden folgende Kindknoten erkannt:

- „**browserToFront**“-Knoten bringen das Browserfenster in den Vordergrund. Dieser Knoten sollte zusammen mit dem „**openUri**“-Knoten verwendet werden, um das Browserfenster zum obersten Fenster zu machen, wenn eine neue Internetseite gezeigt werden soll.
- „**browserResize**“-Knoten erlauben das Ändern der Größe des Browserfensters. Über die Attribute `width` und `height` können die gewünschte Breite und Höhe des Browserfensters angegeben werden.
- „**cancelTest**“-Knoten bricht den Test ab. Die bisherigen Eingaben der Testperson in dieser Aufgabe werden nicht gespeichert. Die Wirkung dieses Knotens entspricht dem Schließen des Aufgabenfensters. Abgebrochene Tests können später durch das Testauswahlfenster an beliebiger Stelle wieder aufgenommen werden.
- „**enable**“-Knoten aktivieren die Komponente, deren Name in den öffnenden und den schließenden Tag eingeschlossen ist.
- „**disable**“-Knoten deaktivieren die Komponente, deren Name in dem Tag angegeben ist.
- „**highlight**“-Knoten heben die angesprochene Komponente hervor, indem diese gelblich unterlegt wird.
- „**unhighlight**“-Knoten stellen die normale graue Farbe der angesprochenen Komponente wieder her.

- „**openUri**“-Knoten öffnen in dem Browser, mit dem das UserTestTool aufgerufen wurde, eine bestimmte Internetseite. Der zu öffnende URI wird in dem öffnenden und schließenden Tag eingeschlossen. Um eine leere Seite zu öffnen, kann „\_blank“ statt eines URIs angegeben werden.
- „**requestFocus**“-Knoten lassen die Eingabemarke zu der Komponente springen, deren Name von dem öffnenden und schließenden Tag eingeschlossen wird.
- „**startNextTask**“-Knoten speichert die Eingaben dieser Aufgabe und startet die nächste Aufgabe, wenn vorhanden. Dieser Knoten sollte der letzte in der Liste der Aktionen sein, da nach ihm keine weiteren Aktionen ausgeführt werden. Sonst würden z. B. danach notierte Stoppuhr-Komponenten nicht gestartet. Auch die letzte Aufgabe sollte durch Druck auf eine Schaltfläche mit diesem Knoten beendet werden, da sonst die Daten der letzten Aufgabe nicht gespeichert würden.
- „**stopWatchStart**“-Knoten starten eine Stoppuhr mit dem im Tag eingeschlossenen Namen. Derartige Stoppuhren sind Aufgabenübergreifend. Somit können auch Zeitspannen über mehrere Aufgaben gemessen werden.
- „**stopWatchElapsed**“-Knoten geben die seit dem Start der Stoppuhr verstrichene Zeit in der Protokolldatei aus. Der Name der zu verwendenden Stoppuhr wird in den Tag eingeschlossen.
- „**stopWatchLapTime**“-Knoten geben die seit dem letzten Aufruf dieses Knotens verstrichene Zeit dieser Stoppuhr in der Protokolldatei aus. Wurde dieser Knoten bisher nicht für diese Stoppuhr aufgerufen, wird die Zeit ausgegeben, die seit dem Start dieser Stoppuhr verstrichen ist. Der Name der zu verwendenden Stoppuhr wird in den Tag eingeschlossen.

### A.3 „dropDownBox“

Der „dropDownBox“-Knoten erzeugt eine ausklappbare Auswahlbox, aus der sich die Testperson einen Text auswählen kann. Jeder Text kann mit einem ganzzahligen Wert versehen werden, der zusammen mit dem ausgewählten Text später in der Protokolldatei erscheint. Die Definition einer solchen Auswahlbox könnte derart aussehen:

```
<dropDownBox name="bewertung" description="Wie finden Sie
das Design der Website?" required="true">
  <item selectable="false" showThisFirst="true">Bitte
wählen Sie</item>
```

```
<item value="5">Schlecht</item>
<item value="10">Geht so</item>
<item value="15">Ganz gut</item>
<item value="20">Echt klasse</item>
</dropDownBox>
```

Folgende Attribute sind möglich:

- „**name**“ weist der Auswahlbox einen Namen zu, über den die Auswahlbox von anderen Komponenten angesprochen werden kann. Dieser Name wird auch in der Protokolldatei angegeben, wenn der ausgewählte Test gespeichert wird.
- „**description**“ legt den einzeiligen Text fest, der über der Auswahlbox angezeigt wird.
- „**required**“ gibt an, ob die Testperson einen selektierbaren (siehe `selectable`) Text auswählen muss (**required="true"**) oder die Aufgabe auch ohne Auswahl eines Textes verlassen darf (**required="false"**).
- „**enabled**“ dient zum Aktivieren oder Deaktivieren der Auswahlbox. Siehe Abschnitt A.1.
- „**highlighted**“ dient zum Hervorheben der Auswahlbox. Siehe Abschnitt A.1.
- „**bottomPadding**“ legt die Anzahl der Pixel fest, die diese Komponente räumlich von der nächsten trennen.

Jeder Texteintrag wird in einen Kindknoten mit dem Namen „**item**“ eingeschlossen. Jeder dieser Knoten kann mit den folgenden Attributen versehen werden:

- „**selectable**“ gibt an, ob dieser Texteintrag eine gültige Auswahl darstellt. Wird **selectable="false"** angegeben, kann der Testbenutzer nicht zur nächsten Seite springen, bis er einen Text in der Auswahlbox gewählt hat, der ohne „**selectable**“-Eintrag oder mit **selectable="true"** definiert wurde. Wird in einem Eintrag **selectable** nicht angegeben, wird **selectable="true"** angenommen.
- „**showThisFirst**“ gibt an, ob dieser Texteintrag als erstes in der Auswahlbox gezeigt wird. Falls es mehr als einen Eintrag gibt, der das Attribut **showThisFirst="true"** hat, wird das zuerst definierte gezeigt.

- „**value**“ erlaubt, jedem Eintrag einen ganzzahligen Wert zuzuweisen. Dieser Wert wird, genau wie der ausgewählte Text, in der Protokolldatei gespeichert. Dies kann für statistische Erhebungen nützlich sein.

#### A.4 „**layout**“

Der „**layout**“-Knoten bezeichnet den Bereich, in dem die Templates untergebracht sind. Darin lassen sich „**template**“-Knoten definieren, die eine oder mehrere Komponenten enthalten können. Diese Komponenten(-gruppen) lassen sich dann in die Aufgaben einbinden.

Ein Beispiel für einen „**layout**“-Bereich könnte etwa so aussehen:

```
<layout>
  <template name="kopf">
    <text name="einText">Oft verwendeter Text</text>
  </template>
</layout>
```

Der „**template**“-Knoten kennt nur das Attribut „**name**“. Hier wird ein Name erwartet, der unter den Templates dieser Aufgabe einmalig ist. Als Kindknoten des „**template**“-Knotens können beliebig viele Komponenten definiert werden.

Weitere Informationen zum Erstellen von Templates und deren Einbindung in die Aufgaben finden Sie im Abschnitt A.7 und A.12.

#### A.5 „**likertScale**“

Die Likert-Skala ist eine bewährte Technik bei der Erstellung von Fragebögen. Sie erlaubt dem Befragten, seine Wertung mit Hilfe von 5 oder 7 Einteilungen abzugeben. Ein Beispiel für das Einbinden einer Likert-Skala sieht folgendermaßen aus:

```
<likertScale name="lesbar" description="Die Seite ist
lesbar" numberOfRatings="7" minLabel="Ich stimme nicht zu"
maxLabel="Ich stimme zu" />
```

Folgende Attribute sind möglich:

- „**name**“ weist der Likert-Skala einen Namen zu, über den die Likert-Skala von anderen Komponenten angesprochen werden kann und der zur Identifikation der Daten in der Protokolldatei dient.

- „**description**“ legt den einzeiligen Text fest, der über der Likert-Skala angezeigt wird.
- „**numberOfRatings**“ gibt die Anzahl der Unterteilungen an, mit der die Likert-Skala gezeigt wird.
- „**minLabel**“ setzt die Beschriftung links von der Likert-Skala. Dies ist meist die minimale Wertung.
- „**maxLabel**“ setzt die Beschriftung rechts von der Likert-Skala. Dies ist meist die maximale Wertung.
- „**required**“ gibt an, ob die Testperson eine Wertung an der Likert-Skala abgeben muss (**required="true"**) oder ob die Aufgabe auch ohne Abgabe einer Wertung verlassen werden darf (**required="false"**).
- „**enabled**“ dient zum Aktivieren oder Deaktivieren der Likert-Skala. Siehe Abschnitt A.1.
- „**highlighted**“ dient zum Hervorheben der Likert-Skala. Siehe Abschnitt A.1.
- „**bottomPadding**“ legt die Anzahl der Pixel fest, die diese Komponente räumlich von der nächsten trennen.

Der „**likertScale**“-Knoten benötigt keine Kindknoten.

## A.6 „**task**“

Dieser Knoten umfasst eine komplette Aufgabe, die der Testperson als ein Fenster dargestellt wird. Jeder „**task**“-Knoten ist Kindknoten des „**userTest**“-Knotens. Die Kindknoten des „**task**“-Knotens, die einzelnen Komponenten, formen das Aussehen des Aufgabenfensters.

Der „**task**“-Knoten erwartet lediglich das Attribut „**name**“. Dieses Attribut weist der Aufgabe einen Namen zu, über den die Aufgabe in der Protokolldatei erkannt werden kann.

## A.7 „**template**“

Templates enthalten Komponenten oder Komponentengruppen, die in den Aufgaben eingebunden werden können. Sie müssen im „**layout**“-Bereich der XML-Datei definiert werden. Diese Knoten kennen lediglich das „**name**“-Attribut, über das sie in den Aufgaben eingebunden werden. Beispiel für ein Template:

```
<template name="kopf">
  <text name="einText">Oft verwendeter Text</text>
</template>
```

Kindknoten des „**template**“-Knotens können eine oder mehrere Komponenten sein, die nach belieben voreingestellt werden können. Beim Einbinden mit Hilfe von „**useTemplate**“ können bereits vorgenommene Einstellungen bei Bedarf wieder überschrieben werden. Weitere Informationen zum Einbinden von Templates finden Sie im Abschnitt 9.12.

## A.8 „**text**“

Dieser Knoten erlaubt das Einfügen von Text in das Aufgabenfenster. Seine einzigen Attribute sind „**name**“ und „**bottomPadding**“. Der auszugebende Text wird zwischen dem öffnenden und dem schließenden „**text**“-Tag notiert. Zur Formatierung können hier viele HTML-Tags benutzt werden, wie `<p/>` und `<h2></h2>`. Auch hier gilt: da es sich um eine XML-Datei handelt, müssen alle Tags, auch die HTML-Tags, geschlossen werden. Ein `<p>` ohne schließendes `</p>` führt zu einer Fehlermeldung beim Start von Scone.

Beispiel eines korrekten „**text**“-Knotens:

```
<text name="einText" bottomPadding="30">
  <h2>Aufgabe 2</h2>
</text>
```

## A.9 „**textField**“

Die Textfeldkomponente erlaubt der Testperson, einen Text einzugeben, der später in der Protokolldatei abgelegt wird. Beispiel eines „**textField**“-Knotens:

```
<textField name="Zeiten" description="Ihr Text" text=""/>
```

Folgende Attribute sind möglich:

- „**name**“ weist dem Textfeld einen Namen zu, über den das Textfeld von anderen Komponenten angesprochen werden kann und der zur Identifikation der Daten in der Protokolldatei dient.

- „**description**“ legt den einzeiligen Text fest, der über dem Textfeld angezeigt wird.
- „**required**“ gibt an, ob die Testperson den Text des Textfeldes verändern muss (**required="true"**), oder ob die Aufgabe auch ohne Änderung des Textes verlassen werden darf (**required="false"**).
- „**text**“ erlaubt, einen Text im Textfeld anzuzeigen.
- „**enabled**“ dient zum Aktivieren oder Deaktivieren des Textfeldes. Siehe Abschnitt A.1.
- „**highlighted**“ dient zum Hervorheben des Textfeldes. Siehe Abschnitt A.1.
- „**bottomPadding**“ legt die Anzahl der Pixel fest, die diese Komponente räumlich von der nächsten trennen.

### A.10 „**title**“

Der Text, der zwischen dem öffnenden und dem schließenden „**title**“-Tag notiert wird, wird für in die Titelleiste des Aufgabenfensters übernommen.

### A.11 „**userTest**“

Der „**userTest**“-Knoten ist das Wurzelement der XML-Testbeschreibungsdatei. Jeder „**task**“- und der „**layout**“-Knoten müssen Kindknoten des „**userTest**“-Knoten sein.

### A.12 „**useTemplate**“

Dieser Knoten dient zum Einbinden von Templates in eine Aufgabe. Es gibt zwei Möglichkeiten, Templates einzubinden: Unveränderndes Einbinden und veränderndes Einbinden.

Beim unverändernden Einbinden wird ein Template so eingebunden, wie es im „**template**“-Knoten definiert wurde. Dazu genügt eine Zeile in der Definition der Aufgabe:

```
<useTemplate name="kopf"/>
```

Die Komponenten, die in dem Template namens „**kopf**“ definiert wurden, werden ohne Änderung übernommen.

Beim verändernden Einbinden werden ebenfalls alle Komponenten aus dem Template übernommen. Es besteht aber die Möglichkeit, beliebig viele der Komponenten in der Aufgabe umzukonfigurieren. Dabei müssen nur die Komponenten erwähnt werden, die auch verändert werden sollen. Alle anderen Komponenten werden, wie im Template angegeben, erstellt. Die Reihenfolge der Komponenten lässt sich durch nachträgliches Konfigurieren nicht verändern. Beispiel für veränderndes Einfügen:

```
<useTemplate name="auswahlfeld">  
  <dropDownBox name="DesignFrage"  
    nameInTemplate="bewertung" description="Wie finden Sie  
    das Design der Website?" />  
</useTemplate>
```

Wenn vorher in einem Template namens „**auswahlfeld**“ eine DropDownBox mit mehreren Einträgen und dem Namen „**bewertung**“ definiert wurde, kann diese nun, wie im obigen Beispiel, in die Aufgabe eingebunden werden. Dabei wird sie um einen eindeutigen Namen und eine Beschriftung ergänzt. Das Attribut „**nameInTemplate**“ ist nötig um genau eine Komponente anzusprechen. Auch wenn es im Template nur eine DropDownBox gibt, muss sie über „**nameInTemplate**“ direkt angesprochen werden.

Der „**useTemplate**“-Knoten kennt nur das „**name**“-Attribut. Es gibt an, nach welchem Template gesucht werden soll.

Alle Komponenten, die verändert werden sollen, müssen als Kindknoten des „**useTemplate**“-Knotens aufgeführt werden. Diese können dann konfiguriert werden. Jede dieser Komponenten muss um das „**nameInTemplate**“-Attribut erweitert werden, das den Namen der Komponente aus dem Template enthalten muss.

Das Attribut „**name**“ der zu ändernden Komponenten enthält den neuen Namen der Komponente. Der neue Name ist nötig, damit im Falle eines mehrfach importierten Templates nicht alle importierten Textfelder den gleichen Namen tragen, wenn sie in der Protokolldatei abgelegt werden.

## B Anhang Einträge der Protokolldatei

In diesem Abschnitt werden alle Einträge beschrieben, die in der Protokolldatei eines Testlaufs stehen können.

### B.1 „userTestResult“

Dieser Knoten ist der Hauptknoten der XML-Protokolldatei. Darunter hängen die Daten aus den einzelnen Aufgaben.

Attribute:

- „**completedTasks**“ gibt die Anzahl der abgearbeiteten Aufgaben an.
- „**numberOfTasks**“ gibt die Anzahl der im Test enthaltenen Aufgaben an.
- „**testPerson**“ enthält den Namen der Testperson.
- „**timeStamp**“ speichert den Zeitpunkt des Testbeginns.
- „**xmlFile**“ gibt an, welche Testbeschreibung für diesen Test verwendet wurde.

### B.2 „task“

Dieser Knoten bezeichnet den Beginn einer neuen Aufgabe. Alle Kind-Knoten sind Aktionen und Daten, die während dieser Aufgabe erfasst wurden.

Attribute:

- „**name**“ gibt den Namen der Aufgabe an.
- „**timeStamp**“ speichert den Zeitpunkt des Aufgabenstarts.

### B.3 „buttonClick“

Die Testperson hat eine Schaltfläche im Aufgabenfenster angeklickt.

Attribute:

- „**name**“ gibt den Namen der Schaltfläche an, auf die geklickt wurde.
- „**action**“ enthält die Aktion der Testperson. Dieser Knoten kennt nur die Aktion „click“.
- „**timeStamp**“ speichert den Zeitpunkt des Anklickens.

### B.4 „browserControl“

Durch das UserTestTool wurde im Browser eine Internetseite aufgerufen. Diese Aktion wurde ausgelöst, als die Testperson eine Schaltfläche angeklickt hat.

Attribute:

- „**frameName**“ gibt den Namen des Browserfensters an.

- **„action** enthält die Aktion der Browsersteuerung. Dieser Knoten kennt nur die Aktionen **„openUri“**, die eine neue Seite im Browser lädt, **„browserToFront“**, die das Browserfenster in den Vordergrund bringt und **„browserResize“**, die die Größe des Browserfensters ändert.
- **„timeStamp“** speichert den Zeitpunkt der Browsersteuerung.
- **„uri“** speichert den aufgerufenen URI.

## B.5 **„browser“**

Der Browser hat eine Aktion gemeldet, z. B. das Laden einer neuen URL durch die Schaltflächen des Browsers.

Attribute:

- **„frameName“** gibt den Namen des Browserfensters an.
- **„action“** enthält die Aktion des Browsers. **„formSubmit“** bezeichnet ein abgeschicktes Formular, **„back“**, **„next“** und **„reload“** die entsprechenden Schaltflächen des Browsers. **„newUrl“** steht für eine von der Testperson eingegebene Internetadresse oder für die Fernsteuerung durch das UserTestTool oder durch JavaScript.
- **„timeStamp“** speichert den Zeitpunkt der Browsersteuerung.
- **„uri“** gibt die neue URL des Browsers an.

## B.6 **„formData“**

Nach dem Abschicken eines Formulars aus einer Internetseite werden die Formulardaten an den Server übermittelt. Diese Daten werden abgefangen. Für jedes Formularfeld wird einer dieser Knoten erzeugt.

- **„frameName“** gibt den Namen des Browserfensters an.
- **„action“** enthält den Namen des Formularfeldes, gefolgt von einem Gleichheitszeichen und dem in das Feld eingetragenen Text.
- **„timeStamp“** speichert den Zeitpunkt der Aktion.
- **„uri“** gibt die neue URL des Browsers an.

## B.7 **„link“**

Der Browser hat das Laden einer neuen URL durch das Anklicken eines Links gemeldet.

Attribute:

- **„frameName“** gibt den Namen des Browserfensters an.
- **„action“** enthält die Aktion, die durch das Anklicken des Links ausgelöst wurde. **„click“** bezeichnet einen Link zu einer anderen Internetseite, **„fragmentOnSamePage“** steht für einen Link, der auf der aktuellen Seite auf ein

bestimmtes Fragment verweist. „samePage“ zeigt einen Link an, der auf die aktuelle Seite verweist.

- „timeStamp“ speichert den Zeitpunkt der Aktion.
- „uri“ gibt die neue URL des Browsers an.

## **B.8 „stopWatch“**

Eine Zeitspanne wurde gemessen.

Attribute:

- „name“ gibt den Namen der Stoppuhr-Komponente an.
- „action“ enthält die Aktion der Stoppuhr. Im Fall von „elapsedTime“ wird die vergangene Zeit seit dem ersten Aufruf eines Timers dieses Namens gespeichert. Bei „lapTime“ wird die Zeitspanne seit dem letzten Aufruf des Timers dieses Namens gespeichert.
- „periodOfTime“ speichert die oben beschriebene Zeitspanne.

## **B.9 „dropDownBoxText“**

Beim Beenden der Aufgabe wurde der Text der DropDownBox ausgelesen, den die Testperson ausgewählt hat.

Attribute:

- „name“ gibt den Namen der DropDownBox an.
- „action“ enthält den Text, den die DropDownBox zuletzt gezeigt hat.
- „timeStamp“ speichert den Zeitpunkt des Auslesens.

## **B.10 „dropDownBoxValue“**

Beim Beenden der Aufgabe wurde der Wert, der dem angezeigten Text der DropDownBox zugewiesen wurde, ausgelesen.

Attribute:

- „name“ gibt den Namen der DropDownBox an.
- „action“ enthält den Wert, der dem zuletzt angezeigten Text zugewiesen ist.
- „timeStamp“ speichert den Zeitpunkt des Auslesens.

## **B.11 „likertScaleSelection“**

Beim Beenden der Aufgabe wird die Bewertung, die die Testperson in der Likert-Skala angeklickt hat, ausgelesen.

Attribute:

- „name“ gibt den Namen der Likert-Skala an.

- „**action**“ enthält die entsprechende Zahl, die die Testperson angeklickt hat. Falls die Testperson keine Wertung über die Likert-Skala abgegeben hat, ist die Zeichenkette leer.
- „**timeStamp**“ speichert den Zeitpunkt des Auslesens.

## **B.12 „textFieldText“**

Beim Beenden der Aufgabe wird der Text des Textfeldes ausgelesen, den die Testperson eingegeben hat.

Attribute:

- „**name**“ gibt den Namen des Textfeldes an.
- „**action**“ enthält den Text, den die Testperson eingegeben hat. Falls die Testperson keinen Text eingegeben hat, enthält dieses Attribut den Text, den das Textfeld angezeigt hat.
- „**timeStamp**“ speichert den Zeitpunkt des Auslesens.