Universität des Saarlandes
Fachrichtung 6.2 Informatik

Fuzz Testing

Zufallstesten von Programmen, Diensten und Handys

Pflichtenheft

Phase	**Phasenverantwortlicher**	**E-Mail**
Pflichtenheft	Stephan Schlicker	stephanschlicker@web.de
Grobentwurf	Holger Hewener	holger@hewener.net
Spezifikation	Andreas Schlicker	andreasschlicker@web.de
Implementierung	Dominik Gummel	bofh@krypt1.cs.uni-sb.de
Testbericht	Sascha Kiefer	sk@intertivity.com
Software	Markus Uhl	markus.uhl@gmx.de

21. Juli 2003

Inhaltsverzeichnis

1 Zielbestimmungen

Mit dem Programm **wmlgen** soll das Erstellen von zufälligen WML-Dokumenten ermöglicht werden. Das Erstellen dieser WML-Dokumente soll mit Konfigurationsdateien durch den Benutzer gesteuert werden können.
Mit diesen zufälligen WML-Dokumenten sollen WAP-Browser auf verschiedenen Endgeräten getestet werden.

1.1 Musskriterien

- Einbinden der DTD, Version 1.3, für WML-Dokumente
- Zufällige Generierung von spezifikationsgetreuen WML-Dokumenten
- Möglicheit zur Konfigurierung des Programms über Konfigurationsdateien
- Validierung der erzeugten WML-Dokumente mit Hilfe eine WML-Validators
- Testen der WML-Dokumente in WAP-Emulatoren
- Speichern von WML-Dokumenten, die einen WAP-Emulator/-Bowser abstürzen lassen, in einem geeigneten Dateiformat
- Integration in Apache-Webserver

1.2 Wunschkriterien

- Finden von Konfigurationen, die WML-Dokuemente erzeugen, welche einen WAP-Browser zum Absturz bringen
- Aufbau einer "Testdatenbank"
- Testen der WML-Dokumente auf WAP-fähigen Endgeräten (z.B. Handys)

1.3 Abgrenzungskriterien

- Für das Programm wird keine graphische Oberfläche entwickelt, es wird ausschließlich über Konsole steuerbar sein.
- Bei der Konfiguration über Konfigurationsdateien werden keine verknüpften, gegenläufigen Nebenbedingungen unterstützt. Ebenso werden keine Maximum-Kriterien unterstützt.
- Es sollen keine Kommentare in das WML-Dokument geschrieben werden.
- Ebenso soll kein zufälliger WML-Skript-Code erzeugt werden.

2 Produkteinsatz

Ein verbreitetes Verfahren, Fehler in Programmen aufzuspüren, ist das Erzeugen zufälliger Eingaben. Anhand einer abstrakten Beschreibung werden zufällige, syntaktisch gültige Eingaben erzeugt (WAP-Seiten) und diese in einem anderen Programm weiterverarbeitet (WAP-Emulator/-Browser).

2.1 Anwendungsbereiche

- Erzeugung von zufälligen WML-Dokumenten
- Testen von WAP-Emulatoren/-Browsern
- Testen der Software-/Ausfallsicherheit von eingebetteten WAP-Microbrowsern (z.B. in Handys)

2.2 Zielgruppe

- Handyhersteller
- Entwickler von WAP-Browsern/-Emulatoren
- Handy-Netzbetreiber
- Forscher der Softwaresicherheit
- Interessierte Handynutzer, die Ihren eigenen WAP-Browser testen wollen

2.3 Betriebsbedingungen

- Büroumgebung

3 Produktumgebung

Das Produkt läuft je nach Einsatzgebiet entweder auf einem Arbeitsplatzrechner oder auf einem Server.

3.1 Software

Emulator-/Browser-Test:

- Windows oder Linux
- WAP/WML Browsersoftware (z.B. Emulator) für das genutzte OS

Test von Endgeräten (eingebette Microbrowser):

- Windows oder Linux (je nach Version)
- Webserver für das genutzte Betriebssystem, der das Abrufen von WML-Dokumenten unterstützt (z.B. entsprechend konfigurierter Apache)

3.2 Hardware

Emulator-/Browser-Test:

- Standard Desktop PC
- min. 5 MB freier Festplattenspeicher
- zusätzlicher Festplattenspeicher zum Abspeichern von erfolgreichen Testdokumenten

Test von Endgeräten (eingebette Microbrowser):

- Server-Computer mit Internetzugang
- WAP-fähiges Endgerät

3.3 Orgware

Test von Endgeräten (eingebette Microbrowser):

- Handy oder Smartphone muss mit SIM-Karte eines Netzbetreibers bestückt sein, welche für Datendienste freigeschaltet ist.
- Webserver muss über Netzwerk-/Internetzugang verfügen.

4 Produktfunktionen

/F100/ Nutzung einer Konfigurationsdatei:
Laden von Konfigirationsvariablen aus einer Datei, um dem Nutzer die Möglichkeit zu geben, die Seitenerstellung zu beeinflussen.

/F110/ Auswahl verschiedener Datenkodierungen für Zeichenketten in der Konfigurationsdatei:

/F111/ Unterstützung von US-ASCII Kodierung

/F112/ Unterstützung von UTF-8 Kodierung

/F113/ Unterstützung von UCS-4 Kodierung

/F114/ Unterstützung von ISO 8859 Kodierung

/F120/ Wahl einer Minimallänge für Zeichensequenzen:
Es besteht die Möglichkeit eine untere Grenze für die Länge aller Zeichenketten anzugeben

/F130/ Wahl einer Durchschnittslänge und deren Standardabweichung für Zeichensequenzen:
Es kann anstelle einer Minimallänge auch eine Durchschnittslänge für Zeichketten angegeben werden, welche noch durch die Angabe einer Standardabweichung variiert werden kann.

/F140/ Selektiver Ausschluss einzelner WML Befehle in der Konfigurationsdatei:
Es können einzelne WML Elemente ausgeschlossen werden, welche nicht in dem erzeugten WML-Dokument auftreten.

/F150/ Regeln speziell für Kinderelemente definierbar:
Der Nutzer kann in der Konfigurationsdatei Regeln definieren, mit denen das Verhalten einzelner WML-Elemente in Abhängigkeit von Ihrem Vater-Elementen bestimmt wird.

/F160/ Grösse der WML-Ausgabedatei durch minimale Grenzen steuerbar:
Der Nutzer kann in der Konfigurationsdatei eine minimale Grösse der Ausgabedatei in Bits angeben. Diese muss von dem erzeugten WML-Dokument überschritten werden.

/F200/ Nutzung der DTD des WML Standards, um alle im Standard spezifizierten Befehle abzudecken:
Hardcodieren aller in der DTD enthaltenen Elemente und deren Attribute.

/F300/ zufällige Generierung des WML Dokumentes aufgrund der Vorgaben in der Konfigurationsdatei

5 Produktdaten

/D100/ Konfigurationsdateien in folgendem Format:
[Sektion1]
Eigenschaft1=Wert1
Eigenschaft2=Wert2
Eigenschaft3=Wert3
...
[Sektion2]
Eigenschaft4=Wert4
Eigenschaft5=Wert5
...

/D200/ Erzeugtes WML Dokument entspricht der WML Spezifikation 1.3

/D300/ Ausgabe des WML Dokumentes durch Voreinstellung auf die Konsole

/D310/ Ausgabe des WML Dokumentes in eine Datei, wenn ein zweiter Kommandozeilenparameter angegeben wird. Nach diesem werden dann die Ausgabedateien benannt. Erstellt werden eine .wml-Datei, welche die Ausgabe Programms darstellt, eine .seed-Datei, welche den Seed des Zufallsgenerators und eine .conf-Datei, welche die benutze Konfiguration beinhaltet.

6 Produktleistungen

/L 100/ Das Produkt sollte jederzeit in der Lage sein, spezifikationskonforme, zufällige WML Dokumente zu erzeugen.

Es werden keine weiteren zeit- oder umfangsbezogenen Anforderungen an die Software gestellt.

7 Benutzeroberfläche

Das Programm besitzt keine graphische Benutzeroberfläche (GUI).
Es wird über die Konsole mit folgenden Möglichkeiten gesteuert:

- Wenn wmlgen keine Parameter übergeben werden, wird eine Standard-Konfigurationsdatei (standard.conf) benutzt. Die Programmausgabe erfolgt auf die Standardausgabe.
- Angabe einer Konfigurationsdatei (*.conf) als Parameter, z.B.
  wmlgen test_ config1.conf
  Programmausgabe erfolgt auf die Standardausgabe.
- Angabe einer Konfigurationsdatei (*.conf) und eines Namens als Parameter z.B.
  wmlgen test_ config1.conf output
  Programmausgabe erfolgt auf die Standardausgabe und zusätzlich werden Ausgabedateien angelegt: output.seed mit dem Seed des Zufallsgenerators und output.wml mit der WML-Ausgabedatei

8 Qualitäts-Zielbestimmungen

Diese Software unterliegt und erfüllt die Qualitätskriterien, die durch die Vorlesung Softwaretechnik I (SS 2003) gestellt wurden.

9 Globale Testszenarien und Testfälle

/T100/ Erzeugung von WML Dokumenten (/F300/)
Das Programm wird ohne Angabe einer Konfiguration gestartet.
Wmlgen soll eine Standard-Konfiguration verwenden und ein gültiges WML-Dokument auf der Standard-Ausgabe ausgeben.

/T200/ Benutzung von Konfigurationsdateien (/F100/)
Wmlgen wird mit einer gültigen Konfigurationsdatei gestartet.
Es soll ein gültiges WML-Dokument ausgegeben werden, das in seiner Zusammensetzung den Angaben in der Konfigurationsdatei entspricht.
Dabei soll das erzeugte Dokument einmal auf der Standard-Ausgabe ausgegeben und einmal in eine Datei gespeichert werden.

/T300/ Funktionstest der einzelnen Konfigurationsmöglichkeiten (/F11*/, /F120/, /F130/, /F140/, /F150/, /F160/)
Bei diesem Test wird wmlgen nacheinander mit speziellen Konfigurationsdateien gestartet, die jeweils nur 1 der Funktionen testen.
Auch hier soll das resultierende WML-Dokument einmal auf der Standard-Ausgabe und in eine Datei ausgegeben werden.

/T400/ Prüfung auf Gültigkeit der WML-Dokumente mit Hilfe des XML-Validators von Elcel Technology (/F200)
(http://www.elcel.com/products/xmlvalid.html)
Bei diesem Test werden verschiedene, mit wmlgen erzeugte, WML-Dokumente auf ihre Konformität bezüglich der WML-DTD (Version 1.3) hin untersucht.

/T500/ Test der erzeugten WML-Dokumente in einem WAP-Emulator
Der WAP-Emulator bekommt als Eingabe ein von wmlgen erzeugtes WML-Dokument.
Es soll hierbei getestet werden, ob der entsprechende WAP-Emulator das spezifikationsgetreue, aber zufällig generierten WML-Dokument korrekt darstellen kann.

/T600/ Testen des Programms über Webserver auf WAP-fähigen Handys (auf vom Kunden bereitgestellten Materialien, Wunschkriterium)
Hierzu wird das Programm wmlgen über ein Perl-Skript auf einem Webserver aufgerufen und liefert ein zufälliges WML-Dokument zurück.

10 Entwicklungsumgebung

10.1 Software

- ANSI C++ kompatibler Compiler: z.B. gcc Version 3.2
- Emacs
- DDD

- WAP-Emulator
- XML-Validator

10.2 Hardware

- Desktop PC

10.3 Orgware

Es ist keine Orgware erforderlich.

11 Ergänzungen

11.1 Programmflussdiagramm

Abbildung 1: Programmflussdiagramm

11.2 Struktur der DTD

Abbildung 2: Baumstruktur der WML-DTD Version 1.3

12 Modifikationen

Im Verlauf des Praktikums können möglicherweise Veränderungen an diesem Dokument erforderlich werden.
Die Modifikationen werden in diesem Abschnitt zusammengefasst.

Version 1.0 Initialversion

A Glossar

- Apache: ein weit verbreiteter Open-Source Web-Server.
- CGI (**C**ommon **G**ateway **I**nterface): Standard für den Informationsaustausch externer Programme oder Gateways, die einen HTTP-Server bedienen.
- WAP (**W**ireless **A**pplication **P**rotocol): Protokoll, das es erlaubt speziell aufbereitete Seiten im Internet über mobile Endgeräte abzurufen
- WML (**W**ireless **M**arkup **L**anguage Specification): Sprache, in der WAP-Seiten geschrieben sind
- XML (E**x**tensible **M**arkup **L**anguage) ist ein Standard des „World Wide Web Consortium“ (W3C) für Internetsprachen. WML ist ein Beispiel für eine solche Sprache. XML ist eine eingeschränkte Teilmenge von SGML (**S**tandardised **G**eneralised **M**arkup **L**anguage; Näheres hierzu siehe Internet).
- DTD (**D**ocument **T**ype **D**efinition): beschreibt die Struktur einer Klasse von SGML- oder XML-Dokumenten mit Hilfe einer Text-Datei, das alle Syntax-Regeln in einem von SGML vorgeschriebenen Format enthält.
- seed: ein Computer kann als deterministisches Gerät keine echten Zufallszahlen erzeugen, sondern nur so genannte Pseudozufallszahlen. Algorithmen zur Erzeugung dieser Pseudozufallszahlen müssen mit einer Zahl (*seed*) initialisiert werden. Bei gleicher seed werden die gleichen Pseudozufallszahlen erzeugt!
- ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange): 7 bit breiter Zeichencode, der keine Sonderzeichen erlaubt (auch bekannt als *ISO-646-US* oder *US-ASCII*).
- Unicode: von der ISO unter den Namen *ISO-10646* standardisiert. Unicode ist ein allumfassender Zeichensatz, der es sich zum Ziel gesetzt hat, die Schwierigkeiten der herkömmlichen, miteinander inkompatiblen Zeichensätze zu überwinden. Unicode ist 32 bit “breit”, d.h. es bietet theoretisch Platz für über 4 Milliarden Zeichen. Aktuell liegt Unicode in der Version 3.1 vor und enthält 94140 verschiedene Zeichen: u.a. lateinische, griechische, kyrillische, arabische, hebräische, japanische und chinesische Zeichen, sogar Runen sowie technische, mathematische und astrologische Sonderzeichen.
  Der 32-bit Zeichenraum von Unicode ist mehrfach in unterschiedlich wichtige Bereiche abgestuft:
  - Die ersten $2^8 = 256$ Zeichen entsprechen genau denen von *ISO-8859-1* (Latin 1).
  - Die ersten $2^{16} = 65536$ Zeichen, welche zusammen *Basic Multilingual Plane (BMP)* genannt werden, enthalten alle Sonderzeichen der ISO-8859 Zeichensätze sowie eine Menge weiterer, wichtiger Zeichen.
  - Die ungebräuchlicheren Zeichen sind in weiteren Ebenen (*planes*) untergebracht.

- UCS-4 Encoding (**U**niversal **C**haracter **S**et) : da Unicode 2^{32} verschiedene Zeichenpositionen hat, benötigt eine naive Kodierung 32 bits = 4 Bytes pro Zeichen.
- UCS-2 Encoding: hier werden nur die ersten 2^{16} Zeichen der BMP mit je 16 bits = 2 Bytes codiert.
- UTF-8 (**U**niversal **T**ransformation **F**ormat): eine wesentlich geschicktere Methode als *UCS-4* / *UCS-2*, Unicode zu codieren. *UTF-8* ist kompatibel zu *ASCII*.
- ISO 8859: enthält die wichtigsten 8-bittigen Zeichensätze. Seine Basis ist das von *ASCII* abgeleitete *ISO-8859-1* oder *Latin-1* (u.a. mit deutschen Sonderzeichen). Die weiteren Varianten enthalten an einigen Positionen andere Sonderzeichen , wie z.B. *ISO-8859-2* (Latin-2) u.a. mit tschechischen, ungarischen und polnischen Zeichen, *ISO-8859-5* (Cyrillic) u.a. mit russischen Zeichen.
- DDD (**D**ata **D**isplay **D**ebugger): Graphische Oberfläche für Kommandozeilendebugger, wie z.B. GDB