Universität des Saarlandes
Fachrichtung 6.2 Informatik

Fuzz Testing

Zufallstesten von Programmen, Diensten und Handys

Implementierungsbericht

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 Implementierung

Das Programm „wmlgen“ wurde gemäß den Vorgaben aus Spezifikation, Grobentwurf und Pflichtenheft implementiert. Die Verteilung der Implementierungsarbeit auf die einzelnen Team-Mitglieder ist in der nachfolgenden Tabelle aufgeführt:

Dominik Gummel	Laden und Speichern von Konfigurationen und wml-Dateien, Attribute, MySQL-Anbindung, Testdatenbank
Holger Hewener	Generierungslogik
Stephan Schlicker	Attribute, DTD
Andreas Schlicker	Attribute
Sascha Kiefer	Konfigurationsdateien parsen, Pseudo-Zufallszahlen
Markus Uhl	Zeichen-Codierung

Testen, Dokumentieren und Fehlersuche wurde von allen Team-Mitgliedern durchgeführt.

1.1 Quellcode

Der Quellcode mit Doxygen-Dokumentation ist zum Abgabezeitpunkt am 21.07. 2003 im CVS mit einem Tag namens “release” gekennzeichnet oder unter foglender Web-Adresse zu finden: `http://alan.cs.uni-sb.de/wml`
Ein `make`-Aufruf übersetzt den Quellcode.

1.2 Web-Version

Unter `http://alan.cs.uni-sb.de/wml/cgi-bin/wmlgen.pl` kann man sich direkt im Web eine WML-Seite von wmlgen generieren lassen.

1.3 Testdatenbank

Unter `http://alan.cs.uni-sb.de/wml/cgi-bin/index.pl` ist unsere Testdatenbank zu erreichen (Login “test”, Passwort “test”).

2 Änderungen des Pflichtenheftes

Es gab keine Änderungen am Pflichtenheft.

3 Änderungen des Grobentwurfes

Es wurden folgende Änderungen am Klassendesign vorgenommen:

3.1 Kontrollstrukturen

3.1.1 Die Klasse WML_Interface

- Änderung: Methode `int get_seed ()` hinzugefügt
- Begründung: Wird gebraucht, um die Seed zum Speichern in einer Datei auslesen zu können.

- Änderung: Attribut geändert
  war: `int seed`
  ist: `dword seed`
  Begründung: Fehler im Grobentwurf; die Seed ist ein 64-bit unsigned long (dword) Wert.

- Änderung: Konstruktor
  `WML_Interface ( std::string, dword _seed )`
  hinzugefügt
  Begründung: Dieser Konstruktor wird gebraucht, um die Seed aus einer Datei einlesen zu können.

- Änderung: Attribut und Methode hinzugefügt
  `std::string timestamp`
  `void setTimestamp ( std::string ts )`
  Begründung: Wird zum Aufbau einer Testdatenbank mit automatischem Testen gebraucht (Kann-Kriterium aus Pflichtenheft).

3.1.2 Die Klasse FileIO

- Änderung: Methoden hinzugefügt
  `void Reset ()`
  `void Close ()`
  Begründung: Schliesst die Datei bzw. setzt den Positionszeiger an den Anfang.

- Änderung: Methode hinzugefügt
  `bool FileIO::Copy (FileIO *source)`
  Begründung: Diese Funktionalität war ursprünglich durch ein fork mit anschliessendem exec-Aufruf von cp realisiert.

- Änderung: Methode hinzugefügt
  `void FileIO::GetCompleteFile (std::string &cf)`
  Begründung: Diese Funktionalität war ursprünglich durch mehrmaliges Aufrufen der Read-Methode realisiert.

3.2 Generierung

3.2.1 Die Klasse Generator

- Änderung: `double rnd()` static deklariert
  Begründung: Auf diese Methode muss auch von Klassen zugegriffen werden, die keinen expliziten Verweis auf den Generator besitzen.

- Änderung: Signatur geändert
  war:
  ```
  Datastructure* eval_attribute
       ( Datastructure::attrtype type )
  ```
  ist:
  ```
  Datastructure* eval_attribute
       ( std::string type )
  ```
  Begründung: Wir lesen aus der Konfigurations-Datei Strings aus; die Konvertierung in eine Instanz eines Datastructure-Objektes ist überflüssig.

- Änderung: Variable `INIFileParser parser` entfernt
  Begründung: Die parse-Methode des Ini-File Parses ist static deklariert.

- Änderung: Standard-Konstruktor entfernt
  Begründung: Singleton-Eigenschaft implementiert.

- Änderung: weiterer Konstruktor als private deklariert
  Begründung: Singleton-Eigenschaft implementiert.

- Änderung: statische Methode CreateInterface hinzugefügt
  Begründung: Singleton-Eigenschaft implementiert
  (um statische Instanz des Generators zu erstellen).

- Änderung: statisches Attribut hinzugefügt
  `int sm_refcount`
  Begründung: Zählt Anzahl der Generator-Referenzen (singleton).

- Änderung: statisches Attribut hinzugefügt
  `Generator* sm_generator`
  Begründung: Speichert statischen Generator-Zeiger (singleton).

- Änderung: Methode `int getUID()` static deklariert
  `Generator* sm_generator`
  Begründung: Es muss global eine eindeutige ID abrufbar sein.

- Änderung: Attribut und Methode hinzugefügt
  `std::string timestamp`
  `void setTimestamp ( std::string ts )`
  Begründung: Wird zum Aufbau einer Testdatenbank mit automatischem
  Testen gebraucht (Kann-Kriterium aus Pflichtenheft).

3.2.2 Die Klasse Rnd

- Änderung: Attribut hinzugefügt
  `static dword m_seed`
  Begründung: Wird zum Speichern der Seed gebraucht.

3.2.3 Die Klasse ConfObject

- Änderung: Methoden hinzugefügt
  ```
  const KeyMap & GetKeymap (const string & section,
       const string & subsection)
  ```
  Begründung: Diese Methode gibt zu einer Section und einer Subsection eine Map
  zurück, die die vorkommenden Keys enthält.

3.2.4 Die Klasse WML_Tree

- Änderung: Änderung der Signatur
  war: `enum encoding`
  ist: `std::string encoding`
  Begründung: Es werden auch benutzerdefinierte encodings unterstützt.

- Änderung: Methode hinzugefügt
  `void addChild ( Element* elem )`
  Begründung: Zur nachträglichen Änderung des repräsentativen Anfangs-Tags.

3.2.5 Die Klasse Element

- Änderung: Attribut `std::string m_tagname` von Klasse Tag hinzugefügt
  Begründung: siehe Tag

- Änderung: pure-virtual Methode hinzugefügt
  `virtual void addChild ( Element* new_elem )`
  Begründung: Wurde hinzugefügt, um Casts zu vermeiden.

- Änderung: Attribut `m_spaces` hinzugefügt
  Begründung: Wird zur Ausgabeformatierung benutzt.

- Änderung: pure virtual Methode
  `std::map<std::string, bool> get_parents ()`
  hinzugefügt
  Begründung: Zur Implementierung von Kindregeln ist es erforderlich, alle Eltern eines Elementes in dieser Map zu speichern.

3.2.6 Die Klasse Tag

- Änderung: Attribut `std::string m_tagname` in die Klasse Element verschoben
  Begründung: Wurde geändert, um Casts zu vermeiden.

- Änderung: Methode `int get_spaces ()` hinzugefügt
  Begründung: Wird zur Ausgabeformatierung benutzt.

- Änderung: Konstruktor geändert
  ```
  Tag ( std::string name, Element* parent,
       std::map< std::string, bool > parents,
       int spaces, bool emptytag = false )
  ```
  Begründung: Der Elementpointer wurde hinzugefügt, um das Erstellen von Kindern passend zu diesem Element zu ermöglichen.

- Änderung: Konstruktor hinzugefügt
  ```
  Tag ( std::string name, Element* parent,
       std::list< std::string > attrlist,
       std::map< std::string, bool > parents,
       int spaces, bool emptytag = false )
  ```
  Begründung: Dieser Konstruktor wurde hinzugefügt, um das Erstellen von Tags ohne vorherige Berechnung von Attributen zu ermöglichen.

3.2.7 Die Klasse PCData

- Änderung: Konstruktor hinzugefügt
  ```
  PCData( std::string encoding, Element* _parent,
       int average_length_string,
       int max_deviation_string)
  ```
  Begründung: Nötig, um Encoding der Zeichen, durchschnittliche String-Länge und maximale Abweichung dieser String-Länge übergeben zu können.

- Änderung: Methoden überladen
  `void addChild ( Element* new_elem )`
  `std::map< std::string, bool > get_parents ()`
  Begründung: Implementierung aufgrund pure-virtual Deklaration in Element nötig.

- Änderung: Methode `int getValidChar( int max_char )` hinzugefügt
  Begründung: Wurde hinzugefügt, um einen in diesem Encoding gültigen Zeichencode zu berechnen.

- Änderung: Attribute hinzugefügt
  `int average_length_string`
  `int max_deviation_string`
  Begründung: Speichert die neu in den Konstruktor aufgenommenen Parameter.

3.3 Attribute

3.3.1 Die Klasse Attribute

- Die Klasse Attribute wird nicht mehr benötigt.
  Im Design war Attribute als Containertyp für Atttribute gedacht, welche in der Klasse Tag in einer Liste gespeichert werden. In der Implementierung werden die Attribute jedoch direkt aus den Strings des Conf-Objects in der generateTree Methode des Generators erstellt. Die Erstellung einer Instanz von Attribute ist hier unnötig, weil die Strings direkt in generateTree abgearbeitet werden.

3.3.2 Die Klasse CData

- Änderung: CData wird jetzt von Datastructure abgeleitet
  (nicht wie ursprünglich vorgesehen von Element).
  Begründung: CData enthält nur Inhalte von Attribute, nicht von Element.

3.3.3 Die Klasse Method

- Änderung: neue Klasse `Method` hinzugefügt
  Begründung: Method ist eine Datenstruktur aus der WML-DTD; sie wurde im Grobentwurf vergessen.

3.4 Hilfsklassen

3.4.1 Die Klasse Debug

- Änderung: Diese Klasse wurde hinzugefügt
  Begründung: Sie dient zur Ausgabe von Fehlermeldungen.

3.4.2 Das Template Ttos

- Änderung: Dieses Template wurde hinzugefügt.
  Begründung: Es konvertiert Datentypen (“T”) zu Strings.

3.4.3 Die Klasse wmlgenDB

- Änderung: Diese Klasse wurde hinzugefügt.
  Begründung: Die Klasse wmlgenDB kapselt den Zugriff auf eine MySQL-Datenbank. Sie wurde nötig, da wir ein Kann-Kriterium des Pflichtenheftes (Aufbau einer Testdatenbank) noch implementiert haben.

4 Änderungen der Spezifikation

Die Testszenarien wurden nicht gändert.