Entwicklung eines Online Webspiel Roboters

We-B-ot
Pflichtenheft

Inhaltsverzeichnis

1 Zielbestimmung

1.1 Einführung

We-B-ot ist ein Bot für Browserspiele, mit dem ein Spieler automatisiert Gegenstände/Ressourcen/ Punkte im Spiel sammeln, kann ohne selbst spielen zu müssen, z.B. um zeitintensive Aktionen automatisiert auszuführen, zu Zeiten an denen der Spieler das Spiel nicht selbst steuern kann.

Da es unmöglich ist einen generischen Bot für alle Browserspiele zu entwickeln, muss die interne Logik des zu spielenden Spiels für jedes Spiel neu geschrieben bzw. angepasst werden. Auch ist es möglich, mit einer entsprechenden Logik den Bot für z.B. Trading-Seiten für Aktien und andere Wertpapiere einzusetzen.

Achtung: Die meisten Onlinespiele verbieten Bots und sperren Spieler, die mit Bots arbeiten. Wir empfehlen, den Bot nur mit Vorsicht einzusetzen. Außerdem sind wir nicht verantwortlich falls ein Account, der diesen Bot einsetzt, gesperrt wird! Die Benutzung des Programms geschieht auf eigene Gefahr!

1.2 Musskriterien

1. Eingabe einer Logik durch den Benutzer
    - Benutzer gibt die gewünschten Funktionen des Bots durch spezielle Sprache über die GUI ein.
2. Aufbauen einer Verbindung zu einer Webseite
    - Es wird ein Browserfenster geöffnet und Formulare an den Server gesendet, in denen die berechneten Werte der Logik enthalten sind und so der gewünschte Spielzug ausgeführt wird.
3. Anwendungsrelevante Daten auslesen
    - Der Bot liest die vom Benutzer vorgegebenen Daten aus der Webseite und verarbeitet sie in der Logik.
4. Verarbeitung der Daten anhand der manuell angegebenen Anwendungslogik
    - Die übergebenen Daten werden anhand der gegebenen Logik bearbeitet.
5. Ausführung der Resultate der Logik durch Senden von Formularen an Server
    - Das Programm sendet über das Browserfenster Formulare an den Server, in denen die berechneten Werte der Logik enthalten sind und führt so den gewünschten Spielzug aus.
6. GUI mit Statusanzeige
    - Spezifische Anzeige der Daten (Punkte für das gegebene Spiel).

1.3 Kannkriterien

1. Tastendruckbefehle ausführen

    - Ermöglicht Portierung des Bots auf Javascript Spiele

1.4 Abgrenzungskriterien

1. Keine optimalen Spielzüge

    - Logiken müssen vom Benutzer spezifiziert werden. Es ist allerdings nicht der Anspruch des Bots, eine perfekte Lösung zum spezifizierten Spiel zu finden. Der Bot ist lediglich als Mittel zum Sammeln von Gegenständen/Ressourcen/Punkten durch stumpfes Wiederholen von einfachen Arbeitsschritten gedacht.

2. Keine Flash-Spiele

3. Keine reinen JavaScript Anwendungen

    - Der Bot ist für die Interaktion mit HTML-Seiten vorgesehen und benötigt diese auch um zu funktionieren.

2 Produkteinsatz

2.1 Anwendungsgebiete

Der Bot ist auf die Interaktion mit HTML-Webseiten ausgelegt. Durch Interfaces kann der Bot auf verschiedene Browserspiele portiert werden. Somit ist das Programm vielseitig einsetzbar. Der Haupteinsatzzweck ist also die automatische Bedienung von Browserspielen zum Sammeln von Gegenständen/Ressourcen/Punkten.

2.2 Zielgruppe

Die Zielgruppe des Bots sind Spieler von Browserspiele. Je nach angegebener Logik kann der Bot vielseitig eingesetzt werden: Als „Farmbot“, um möglichst ohne Aufwand an Gegenstände/Ressourcen/Punkte zu kommen, oder als „Tradebot“, um zu unterschiedlichsten Zeiten auf Ereignisse, z.B. Handelsangebote, innerhalb des spezifizierten Spiels zu reagieren.

2.3 Ort

Der Bot kann von jedem beliebigen Windows Computer aus gestartet werden. Damit der Bot auch z.B. nachts oder in Abwesenheit des Spielers arbeiten kann, muss der PC, auf dem der Bot gestartet wurde, eingeschaltet bleiben. Eine aktive Internetverbindung wird ebenfalls benötigt, da der Bot sonst keine Verbindung zur HTML-Seite aufbauen kann.
Da der Bot in einem aktiven Browserfenster arbeitet, ist die Verarbeitung von Cookies dem Browser überlassen. Der Bot kann auf allen Webseiten eingesetzt werden, die den gewählten Browser unterstützen.

3 Produktumgebung

3.1 Benötigte Hardware

Für den We-B-ot wird ein Computer mit Windows und Internetverbindung benötigt. Weitere Forderungen an die Hardware werden nicht gestellt.

3.2 Software

Damit das Programm ausgeführt werden kann, muss eine aktuelle Java-Installation (Compile-Version Java 1.8) vorliegen.
Zum Ausführen wird außerdem **einer** der folgenden Browser benötigt:

- Mozilla Firefox (empfohlener Browser)
- Google Chrome
  - benötigt Zusatzsoftware “chromeDriver.exe”
  - `https://sites.google.com/a/chromium.org/chromedriver/downloads`
- Microsoft Internet Explorer
  - benötigt Zusatzsoftware “ieDriver.exe”
  - `https://code.google.com/p/selenium/downloads/list`

4 Funktionalität

F/100/ Webseiten müssen geparst werden können. Dies beinhaltet Verbindungsaufbau zu einer Webseite und parsen der Antwort zur weiteren Verarbeitung.

F/1000/ HTTP Code der Webseite muss ausgelesen werden können.

F/101/ HTTP Dokumente müssen nach benötigten Daten untersucht werden können.

F/1010/ Das Programm muss HTTP Dokumente nach benötigten Elementen untersuchen und deren Inhalt zurückgeben können. (benötigt **F/103/**)

F/1011/ Ein HTTP Dokument muss auf alle beinhalteten Formulare untersucht werden können. Es müssen alle benötigten Daten zum Ausfüllen dieser Formulare zurückgegeben werden können (benötigt **F/103/** zum Bestimmen der relevanten Formulare).

F/102/ Es müssen Formulare an eine Webseite gesendet werden können.

F/1020/ Die Formulare müssen mit den benötigten Werten und Benutzereingaben ausgefüllt und abgesendet werden können. (benötigt **F/103/** bei benötigten Zusatzdaten, wie Passwörtern, etc.)

F/1021/ (Optional) Senden von Javascript Keypresses an Webseite, um JavaScript-basierte Spiele bedienen zu können.

F/103/ Ein Benutzer muss über eine GUI das Verhalten des Bots, sowie die benötigten Daten für die Interaktion von Bot und Webseite, in Pseudocode eingeben können.

F/1030/ Der Code muss in vom Bot verwendbare Logik-Instruktionen umgeschrieben werden. Die Syntax der dafür verwendeten Sprache ist in Abschnitt 5 spezifiziert.

F/1031/ Der Code muss in vom Bot verwendbare Logik-Instruktionen umgesetzt werden.

F/104/ Der Bot muss den Benutzer bei Angabe einer E-Mail-Adresse im Falle eines Fehlers benachrichtigen können.

F/1040/ Senden einer E-Mail mit Ausgabe der Fehlermeldung.

F/1041/ Bei Existenz von Benutzername und Passwort, senden der URL und bittet um manuelle Überprüfung des Accounts.

F/105/ Es wird eine GUI benötigt, die über Kontrollfunktionen, Statusanzeige, sowie ein Eingabefeld für die Logik für **F/103/** verfügt.

F/1050/ Das Programm muss über einen Button der GUI gestoppt werden können.

F/1051/ Das Programm muss über einen Button der GUI gestartet werden können.

F/1052/ Es muss ein Textfeld oder Button zum Laden einer Spiellogik existieren, aus dem bei Druck des Start-Buttons eine neue Spielelogik nach **F/103/** ausgelesen wird.

F/1053/ Es muss eine Statusanzeige existieren, die den momentanen Zustand des Programms anzeigt. Zustände: Verbunden, Inaktiv, Keine Logik, Verarbeite Logik.

F/1054/ (Optional) Anzeige eines bestimmten Wertes, spezifiziert nach **F/104/**

F/106/ Ein Logger muss die Abläufe des Hauptprogramms protokollieren können.

F/1060/ Die Abläufe des Bots werden überwacht und protokolliert.

F/1061/ Der Logger muss das Protokoll in einer Textdatei speichern können. Diese werden im Ordner gespeichert, von dem der User den Bot startet.

5 Spezifizierung der Sprache

Die Sprache unterstützt folgende Eingaben. Die Werte in ' sind Platzhalter für das Format, die ' müssen aber mit angegeben werden. Werte in * sind Platzhalter für Namen (hier sind keine ' oder * anzugeben). Ein Beispiel mit allen befehlen folgt am Ende. Für die Sprache wurde Xtext 2.8.0 verwendet.

- start 'dd.MM.YYYY hh:mm:ss'
  - legt den Startzeitpunkt fest
  - wenn nicht richtig angegeben, startet das Programm sofort
  - folgende Schreibweisen erlaubt: start, Start, START
- email 'ab@c.de'
  - legt die E-Mail Adresse für Benachrichtigungen fest
  - optional
  - folgende Schreibweisen erlaubt: E-Mail, Address, EMail, Mail, email, mail, address
- connect 'url'
  - verbindet mit angegebener URL
  - folgende Schreibweisen erlaubt: connect, Connect
- read at 'xpath'
  - liest den Wert am gegebenen xpath aus
  - folgende Schreibweisen erlaubt: read at, readat, Read At, readAt, ReadAt
- write 'text' at 'xpath'
  - schreibt den Wert von text an den gegebenen xpath
  - folgende Schreibweisen sind erlaubt: write, Write und at, At, AT jeweils frei kombiniert
- click at 'xpath'
  - klickt auf das Element am gegebenen xpath
  - folgende Schreibweisen erlaubt: click at, clickat, clickAt, Click At, ClickAt
- var *VarName* = read At 'xpath'
  - setzt die Varibale mit *VarName* auf den Wert von "read At"
  - folgende Schreibweisen erlaubt: Var, var, VAR
- wait 'millis'
  - lässt das Programm so viele Millisekunden warten, wie der Wert in 'millis'
  - folgende Schreibweisen erlaubt: wait, Wait

- show 'name' at 'xpath'
  - zeigt den Wert des Xpath mit den gegebenen Namen in einem Teil der GUI an. Ist als Liste möglich
- notify 'message'
  - sendet eine Nachricht an den Nutzer. Funktioniert nur wenn email angegeben wurde
  - folgende Schreibweisen erlaubt: notify, Notify
- noClose
  - beendet den Browser nach Ablauf des Skripts nicht
  - folgende Schreibweisen erlaubt: not close, no close, no Close, notclose, notClose, noclose, noClose, NotClose, NoClose
- stop
  - zeigt das Ende des Skripts an
  - folgende Schreibweisen erlaubt: stop, Stop, STOP
- if *bedingung* then *aktionen* else *aktionen* endif
  - besteht aus if, If oder IF
  - bedingungen können mit >, <, >=, <=, =, ==, != verglichen werden, außerdem können die Eingaben mit +, -, * und / verarbeitet werden
  - bedinungen können mit |, ||, OR, or, Or, &, &&, AND, and, And verknüpft werden
  - enthält then, Then, THEN
  - enthält Optional else, Else, ELSE
  - aktionen können alles bisher genannte, inklusive while-Schleifen sein.
  - wird geschlossen mit endif, EndIf, Endif, ENDIF geschlossen
- while *bedingung* loop *aktionen* endloop
  - Besteht aus while, While, WHILE,
  - Bedingungen können mit >, <, >=, <=, =, ==, != verglichen werden, außerdem können die Eingaben mit +, -, * und / verarbbeitet werden
  - Bedinungen können mit |, ||, OR, or, Or, &, &&, AND, and, And verknüpft werden
  - enthält Loop, loop, LOOP
  - Aktionen können alles bisher genannte sein.
  - wird geschlossen mit endloop, Endloop, EndLoop, ENDLOOP

```
start '03.07.2015 18:38:00'
email 'swp2015mail@gmail.com'
noClose

Connect 'https://tribalwars.net '
Write 'swp2015' at '//*[@id="user"]'
Write '[redacted]' at '//*[@id="password"]'
Click At '/html/body/div[2]/div[1]/div[2]/div/div[2]/div[2]/form/div/div/div/a'
Click At '/html/body/div[2]/div[1]/div[2]/div/div[2]/div[2]/div[1]/div[2]/form/div[1]/div/a'

show wood iron at '//*[@id="wood"]' '//*[@id="iron"]'

Connect 'https://en80.tribalwars.net/game.php?village=73059&screen=place'
var speer = read at '//*[@id="units_entry_all_spear"]'
var axe = read at '//*[@id="units_entry_all_axe"]'

while speer >= '(30)' & axe >= '(10)'
loop
write '29' at '//*[@id="unit_input_spear"]' write '9' at '//*[@id="unit_input_axe"]'
write '597|729' at '/html/body/table/tbody/tr[2]/td[2]/table[2]/tbody/tr/td/table/tbody/tr
                    /td/table/tbody/tr/td/form/div[1]/table/tbody/tr[1]/td/div[2]/input'
click at '//*[@id="target_attack"]'
click at '//*[@id="troop_confirm_go"]'
notify 'angriff gesendet'
var speer = read at '//*[@id="units_entry_all_spear"]'
var axe = read at '//*[@id="units_entry_all_axe"]'
endloop

notify 'Keine weiteren Angriffe'

var iron = read at '//*[@id="iron"]'

if iron > '100'
then notify "genug Eisen da"
else notify "du brauchst mehr eisen"
endif

STOP
```

Abbildung 1: Beispiel für eine .bla-Datei

6 Daten

6.1 Benötigte Daten

- Logikdatei (*.bla)
  - erstellt nach Spezifikation der Sprache (siehe 5)

6.2 Eingaben

- Browserauswahl
  - für Internet Explorer und Chrome Dateipfad für ieDriver bzw. chromeDriver

7 Aufbau und GUI

7.1 Klassendiagramm

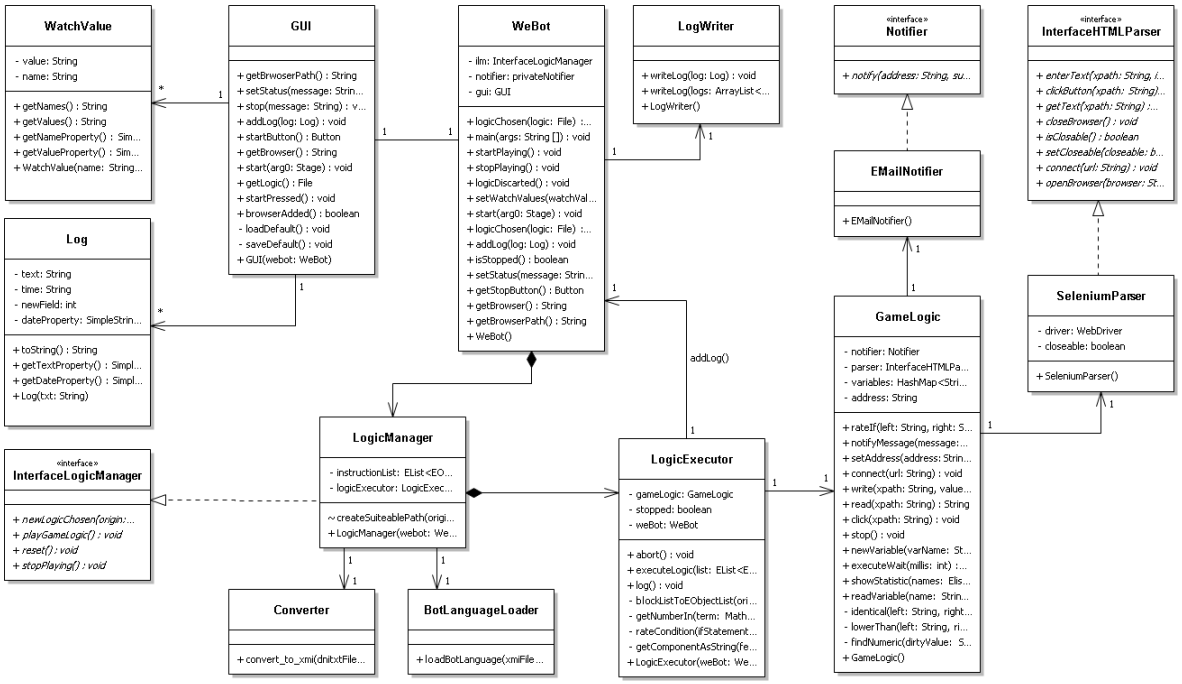

Abbildung 2: Klassendiagramm

Der SeleniumParser und der EMailNotifier können blockieren, deshalb sollten diese in einem extra Thread getrennt vom GUI Thread laufen. Damit wird die GUI nicht blockiert. Die Klasse GameLogic führt nur Befehle aus, LogicExecutor sorgt für den Konntrollfluss. Daher sollten alle 4 Klassen in einem gemeinsamen Thread laufen. Jedoch müssen während der Ausführung des Programms Ausgaben (also Logs/WatchValues) produtiert werden. Deshalb ist die addLog() Methode von LogicExecutor zum WeBot wichtig. Umweg über LogicManager wäre in diesem Fall unnötig.

7.2 Architekturdiagramm

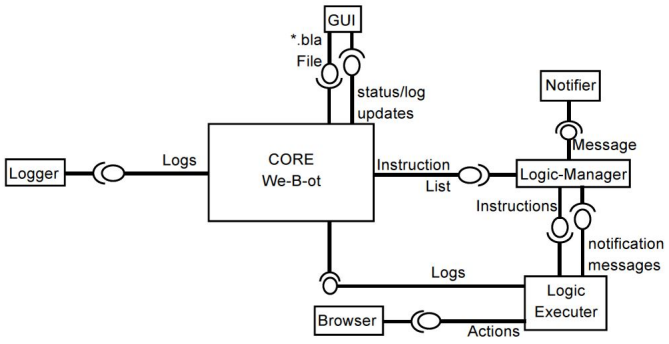

Abbildung 3: Architekturdiagramm

7.3 GUI

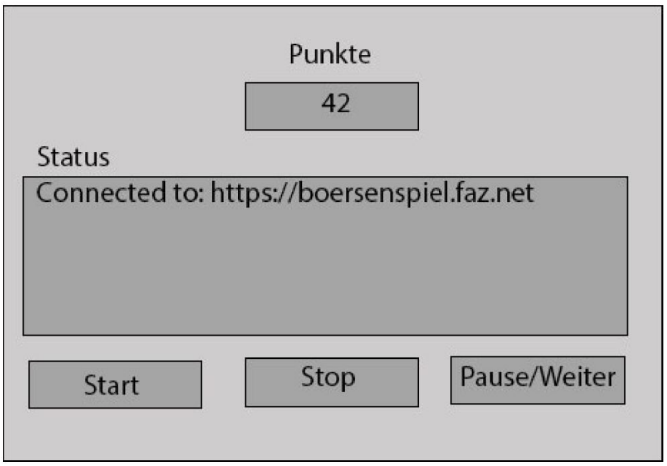

Abbildung 4: Entwurf der GUI

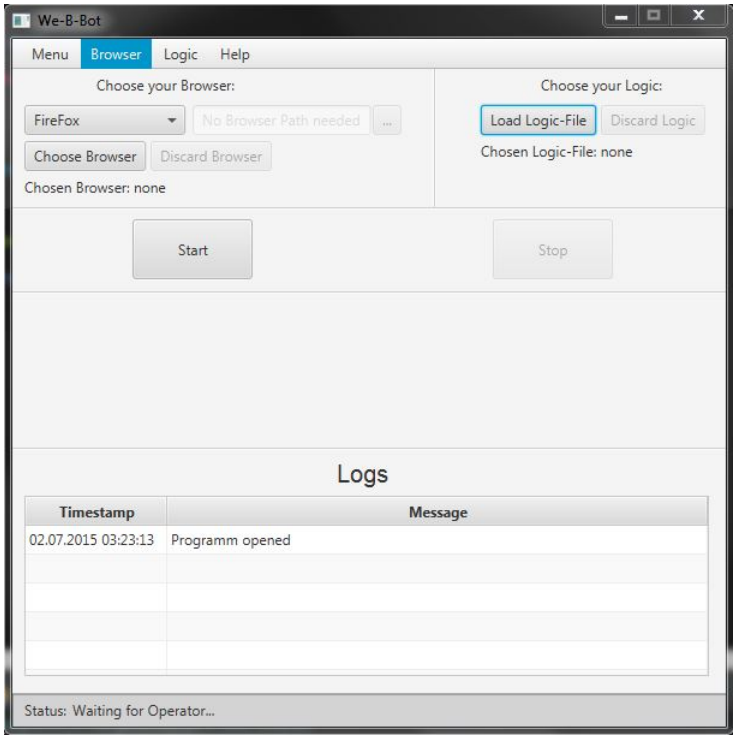

Abbildung 5: GUI nach Start

8 Tests

Folgende Tests wurden durchgeführt:

1. Nicht erlaubte Eingabe in Xtext bzw. *.bla-Datei

    - Anstelle von “read at” z.B. “REaD at”, oder nicht existenten Befehl “Read”
      Die Eingabe wurde ignoriert. Es wurde kein Fehler angezeigt und keine Exception geworfen.

2. Nicht-existierende Xpath

    - Eingabe eines ungültigen Xpath, der auf der Seite nicht existiert
      Es entstand eine Fehlermeldung bzw. Exception, da der Xpath der als nächstes angesteuert werden sollte, nicht gefunden wurde.

3. Captchas

    - Es tritt unerwartet ein Captcha auf
      Es entstand eine Fehlermeldung bzw. Exception, da der Xpath der als nächstes angesteuert werden sollte, nicht gefunden wurde.

4. Unterbrochene Internetverbindung

    - Lan-Kabel wurde gezogen oder W-Lan Adapter wurde deaktiviert
      Es entstand eine Fehlermeldung bzw. Exception, da der Xpath der als nächstes angesteuert werden sollte, nicht gefunden wurde.

5. Browser wird geschlossen

    - Das Browserfenster wird mitten im Betrieb geschlossen
      Der Bot wird angehalten. User muss Stop-Button allerdings manuell betätigen, um den Bot neu starten zu können.

6. Start/Stop/Start

    - Bevor das Browserfenster geöffnet wird, wird Stop gedrück, kurz darauf wieder Start
      Der Bot wird noch das Browserfenster für die alte Logik öffnen und dann beide Logiken anhalten.

9 Nicht-funktionale Eigenschaften

Der Bot soll in angemessener Zeit das Browserspiel steuern. Weitere Laufzeitbeschränkungen sind nicht gefordert.

10 Verbesserungen

1. Kannkriterien

   Die in den Kannkriterien beschriebenen KeyPresses könnten noch hinzugefügt werden.

2. Erweiterung des Befehls “notify”

   Der Befehl “notify” könnte so erweitert werden, dass neben einem Nachrichtenstring auch der Wert einer oder mehrerer Variablen mit angegeben werden kann.

3. Erweiterung des Befehls “var”

   Variablen so erweitern, dass Variablen nicht nur mit “read at” sondern auch durch andere Variablen und mathematische Operationen von Variablen entstehen können.

4. Dynamische Größe der GUI

   Die GUI könnte resizable gemacht werden. Dafür müssten sämtliche X und Y-Werte für GUI-Elemente in Abhängigkeit von der Größe der GUI gesetzt werden.

5. Erweiterung von If

   Die If-Abfrage durch ein “else if” erweitern (Kann durch ein eigenes “if” im else-Block erreicht werden, allerdings müssen dann auch zwei “end if” stehen).

6. Neuer Befehl “Log”

   Ein Befehl, der ein bestimmtes Ereignis logt, ohne es explizit E-Mails zu versenden.