

Validierung

Simon Bischof, Jan Haag, Adrian Herrmann, Lin Jin, Tobias Schlumberger, Matthias Schnetz

Praxis der Softwareentwicklung

Projekt 3:

Automatisches Prüfen der Korrektheit von Programmen

Gruppe 1



WS 2011/2012

Inhaltsverzeichnis

1	GUI Testplan	3
1.1	Menubar	3
1.1.1	„File“ → „New“	3
1.1.2	„File“ → „Load“	3
1.1.3	„File“ → „Save“	3
1.1.4	„File“ → „Exit“	4
1.1.5	„Edit“ → „Undo“	4
1.1.6	„Edit“ → „Redo“	5
1.1.7	„Edit“ → „Cut“	5
1.1.8	„Edit“ → „Copy“	6
1.1.9	„Edit“ → „Paste“	6
1.1.10	„Edit“ → „Settings“	6
1.1.11	„Run“ → „Run“	6
1.1.12	„Run“ → „Single Step“	6
1.1.13	„Run“ → „Random Test“	6
1.1.14	„Run“ → „Verify“	7
1.1.15	„Help“ → „Help“	7
1.1.16	„Help“ → „About“	7
1.2	Frames	7
1.2.1	Settingsframe	7
1.2.2	Randomtestframe	7
1.2.3	Helpframe	8
1.3	Views	8
1.3.1	Editor	8
1.3.2	Globalbreakpointview	9
1.3.3	Helpbox	9
1.4	Buttons	9
1.4.1	CheckSyntax	9
1.4.2	Run	9
1.4.3	Single Step	9
1.4.4	Pause	9
1.4.5	Stop	9
1.4.6	Validate	9

1 GUI Testplan

1.1 Menubar

1.1.1 „File“ → „New“

1. Öffnen einer neuen Datei
 - Erwartetes Ereignis: Der Inhalt des Editors wird ohne zu speichern gelöscht, Breakpoints werden entfernt und die Konsolen geleert.
 - Status: **FEHLSCHLAG**
Breakpoints werden nicht entfernt, Konsole nicht geleert

1.1.2 „File“ → „Load“

1. Laden einer nichtexistenten Datei
 - Erwartetes Ereignis: Die Datei wird nicht geladen.
 - Status: **OK**
2. Laden einer Datei, die vom Programm erzeugt wurde
 - Erwartetes Ereignis: Die ausgewählte Datei wird in den Editor geladen.
 - Status: **OK**
3. Laden einer reinen Textdatei (zum Beispiel: txt, tex)
 - Erwartetes Ereignis: Die ausgewählte Datei wird in den Editor geladen.
 - Status: **OK**
4. Laden einer durch Textbearbeitungsprogramme erstellten Textdatei (Getestet: docx, pdf)
 - Erwartetes Ereignis: Die Datei wird nicht geladen.
 - Status: **FEHLSCHLAG**
docx-Dateien werden geladen, Programm hängt sich auf bei pdf-Dateien
5. Laden einer Nichttextdatei (zum Beispiel: exe)
 - Erwartetes Ereignis: Die Datei wird nicht geladen.
 - Status: **FEHLSCHLAG**
Programm hängt sich auf
6. Nachdem eine Datei korrekt geladen wurde
 - Erwartetes Ereignis: Der alte Inhalt des Editors wird ohne zu speichern gelöscht, Breakpoints werden entfernt und die Konsolen geleert.
 - Status: **FEHLSCHLAG**
Breakpoints werden nicht entfernt, Konsole nicht geleert

1.1.3 „File“ → „Save“

1. Speichern in einer nichtexistenten Datei
 - Erwartetes Ereignis: Die Datei wird erzeugt, der Inhalt des Editors darin gespeichert.
 - Status: **OK**
2. Speichern in einer vom Programm erzeugten Datei
 - Erwartetes Ereignis: Der Inhalt der Datei wird durch den des Editors ersetzt.

- Status: OK
3. Speichern in einer reinen Textdatei
 - Erwartetes Ereignis: Der Inhalt der Datei wird durch den des Editors ersetzt.
 - Status: OK
 4. Speichern in einer durch Textbearbeitungsprogramme erstellten Textdatei (zum Beispiel: docx, pdf)
 - Erwartetes Ereignis: Die Datei wird nicht gespeichert.
 - Status: FEHLSCHLAG
Die Datei wird gespeichert.
 5. Speichern in einer Nichttextdatei (zum Beispiel: exe)
 - Erwartetes Ereignis: Die Datei wird nicht gespeichert.
 - Status: FEHLSCHLAG
Die Datei wird gespeichert.

1.1.4 „File“ → „Exit“

1. Beenden des Programms
 - Erwartetes Ereignis: Das Programm wird sofort beendet.
 - Status: OK

1.1.5 „Edit“ → „Undo“

1. Rückgängigmachen des zuletzt eingetippten Zeichen
 - Erwartetes Ereignis: Das zuletzt eingetippte Zeichen wird gelöscht.
 - Status: OK
2. Beliebige Wiederholung von Punkt 1
 - Erwartetes Ereignis: Die zuletzt eingetippten Zeichen werden gelöscht.
 - Status: OK
3. Rückgängigmachen des zuletzt gelöschten Zeichen
 - Erwartetes Ereignis: Das zuletzt gelöschte Zeichen wird wieder hergestellt.
 - Status: FEHLSCHLAG
Punkt 4 funktioniert bis auf das zuerst gelöschte Zeichen
4. Beliebige Wiederholung von Punkt 3
 - Erwartetes Ereignis: Die zuletzt gelöschten Zeichen werden wieder hergestellt.
 - Status: OK
5. Rückgängigmachen der letzten Paste-Aktion
 - Erwartetes Ereignis: Die zuletzt eingefügte Zeichenkette wird gelöscht.
 - Status: OK
6. Beliebige Wiederholung von Punkt 5
 - Erwartetes Ereignis: Die zuletzt eingefügten Zeichenketten werden gelöscht.
 - Status: OK
7. Rückgängigmachen der letzten Cut-Aktion

- Erwartetes Ereignis: Die zuletzt gelöschte Zeichenkette wird wieder hergestellt.
- Status: **OK**

8. Beliebige Wiederholung von Punkt 7

- Erwartetes Ereignis: Die zuletzt gelöschten Zeichenketten werden wieder hergestellt.
- Status: **OK**

9. Rückgängigmachen der Funktion „File“ → „New“

- Erwartetes Ereignis: Der alte Inhalt des Editors wird wieder hergestellt.
- Status: **OK**

10. Punkt 9 nachdem mehrmals auf „File“ → „New“ geklickt wurde

- Erwartetes Ereignis: Der alte Inhalt des Editors wird wieder hergestellt.
- Status: **FEHLSCHLAG**
Das erwartete Ereignis passiert erst nach dem zweiten Undo

11. Rückgängigmachen der Funktion „File“ → „Load“

- Erwartetes Ereignis: Der alte Inhalt des Editors wird wieder hergestellt.
- Status: **OK**

12. Punkt 11 nachdem mehrmals „File“ → „Load“ benutzt wurde

- Erwartetes Ereignis: Der Inhalt der zuletzt geladenen Datei wird im Editor hergestellt.
- Status: **FEHLSCHLAG**
Der Inhalt vor den Load-Vorgängen wird nach dem zweiten Undo hergestellt

13. Undo, obwohl noch keine Aktion aufgeführt wurde

- Erwartetes Ereignis: Es passiert nichts.
- Status: **OK**

1.1.1.6 „Edit“ → „Redo“

1. Rückgängigmachen der letzten Undo-Aktion

- Erwartetes Ereignis: Die rückgängig gemachte Aktion wird hergestellt.
- Status: **OK**

2. Beliebige Wiederholung von Punkt 1

- Erwartetes Ereignis: Die rückgängig gemachten Aktionen werden hergestellt.
- Status: **OK**

1.1.1.7 „Edit“ → „Cut“

1. Löschen der markierten Zeichenkette

- Erwartetes Ereignis: Die markierte Zeichenkette wird gelöscht.
- Status: **OK**

2. Cut ohne markierte Zeichenkette

- Erwartetes Ereignis: Es passiert nichts.
- Status: **FEHLSCHLAG**
Programm stürzt ab.

1.1.8 „Edit“ → „Copy“

1. Kopieren der markierten Zeichenkette
 - Erwartetes Ereignis: Die markierte Zeichenkette wird zum Kopieren gespeichert.
 - Status: **OK**
2. Beliebige Wiederholung von Punkt 1
 - Erwartetes Ereignis: Die zuletzt kopierte Zeichenkette wird gespeichert.
 - Status: **OK**
3. Copy ohne markierte Zeichenkette
 - Erwartetes Ereignis: Es passiert nichts.
 - Status: **FEHLSCHLAG**
Programm stürzt ab.

1.1.9 „Edit“ → „Paste“

1. Einfügen der aus dem Programm kopierten Zeichenkette
 - Erwartetes Ereignis: Die kopierte Zeichenkette wird im Editor eingefügt.
 - Status: **OK**
2. Einfügen der aus einem anderen Programm kopierten Zeichenkette
 - Erwartetes Ereignis: Die kopierte Zeichenkette wird im Editor eingefügt.
 - Status: **OK**
3. Beliebige Wiederholung von Punkt 1 oder 2
 - Erwartetes Ereignis: Die kopierte Zeichenkette wird jedes Mal im Editor eingefügt.
 - Status: **OK**

1.1.10 „Edit“ → „Settings“

1. Öffnen des Settingsfensters
 - Erwartetes Ereignis: Das Fenster zur Einstellung von Z3-Settings wird geöffnet.
 - Status: **OK**

1.1.11 „Run“ → „Run“

Siehe Abschnitt 1.4.2

1.1.12 „Run“ → „Single Step“

Siehe Abschnitt 1.4.3

1.1.13 „Run“ → „Random Test“

1. Öffnen des Randomtestfensters
 - Erwartetes Ereignis: Das Fenster für Randomtests wird geöffnet.
 - Status: **OK**

1.1.14 „Run“ → „Verify“

Siehe Abschnitt 1.4.6

1.1.15 „Help“ → „Help“

1. Öffnen des Helpfensters und Anzeigen der Helpdokumentation
 - Erwartetes Ereignis: Die Helpdokumentation wird geöffnet.
 - Status: OK

1.1.16 „Help“ → „About“

1. Öffnen des Aboutfensters
 - Erwartetes Ereignis: Das Aboutfenster wird geöffnet.
 - Status: OK

1.2 Frames

1.2.1 Settingsframe

1. Speichern von korrekten Eingaben
 - Erwartetes Ereignis: Es wird eine Erfolgsmeldung ausgegeben und die neuen Eingaben stehen in den entsprechenden Textfeldern.
 - Status: OK
2. Speichern von inkorrekten Eingaben
 - Erwartetes Ereignis: Es wird eine Fehlermeldung ausgegeben und die alten Werte werden wiederhergestellt.
 - Status: BEHOBEN
Wenn der Pfad nicht korrekt eingegeben wurde, wird trotzdem die Erfolgsmeldung angezeigt.
3. Klick auf „Close“ Button
 - Erwartetes Ereignis: Das Settingsframe wird geschlossen.
 - Status: OK
4. Ausführen von 1, 3 und anschließendes Öffnen des Fensters.
 - Erwartetes Ereignis: Die bei 1 eingegebenen neuen Werte stehen immernoch in den entsprechenden Textfeldern.
 - Status: OK

1.2.2 Randomtestframe

1. Öffnen bei einem syntaktisch inkorrekten Programm
 - Erwartetes Ereignis: Es erscheint die Meldung im Randomtestfenster und in der Errorkonsole, dass das Programm syntaktisch inkorrekt ist.
 - Status: OK
2. Öffnen bei einem Programm ohne Parameter
 - Erwartetes Ereignis: Es erscheint die Meldung im Randomtestfenster, dass das Programm keine Parameter hat.

- Status: OK

3. Öffnen bei einem Programm mit Parameter

- Erwartetes Ereignis: Der Benutzer hat nun die Möglichkeit, Anzahl der Tests und Werte für die Parameter einzugeben. Werte, die für die Variable nicht sinnvoll sind, werden grau angezeigt und sind nicht editierbar.
- Status: OK

4. Durchführung der Tests mit korrekten Eingaben

- Erwartetes Ereignis: Die Tests werden durchgeführt und das Ergebnis wird zusammen mit den zufällig ausgewählten Werten in der Miskonsole angezeigt.
- Status: OK

5. Durchführung der Tests mit inkorrekten Eingaben

- Erwartetes Ereignis: Falls die Anzahl der Tests eine ungültige Eingabe oder kleiner als 1 ist, wird kein Test durchgeführt. Ansonsten werden ungültige Werte durch „0“ bzw. „false“ ersetzt. Das Ergebnis wird zusammen mit den zufällig ausgewählten Werten in der Miskonsole angezeigt.
- Status: OK

1.2.3 Helpframe

1. Auswählen der einzelnen Abschnitte

- Erwartetes Ereignis: Der ausgewählte Abschnitt wird angezeigt.
- Status: OK

1.3 Views

1.3.1 Editor

1. Eingabe, Modifikation von Quelltext im idle-Zustand

- Erwartetes Ereignis: Der Inhalt des Editors kann beliebig verändert werden, solange es kein Programm läuft oder pausiert ist.
- Status: OK

2. Eingabe, Modifikation von Quelltext im nicht-idle-Zustand

- Erwartetes Ereignis: Der Inhalt des Editors kann nicht verändert werden, solange ein Programm läuft oder pausiert ist.
- Status: OK

3. Syntaxhighlighting

- Erwartetes Ereignis: Die Keywords „int, bool, array, true, false, main, while, if, else, return, assert, assume, ensure, invariant“ und Zahlen werden farbig hervorgehoben.
- Status: OK

4. Setzen oder Entfernen von Statementbreakpoints im nicht-idle-Zustand

- Erwartetes Ereignis: Breakpoints können nicht gesetzt oder entfernt werden, solange ein Programm läuft oder pausiert ist.
- Status: OK

5. Setzen von Statementbreakpoints im idle-Zustand

- Erwartetes Ereignis: Statementbreakpoints können nur gesetzt werden, wenn in der Zeile ein Statement steht.
- Status: **OK**

6. Entfernen von Statementbreakpoints im idle-Zustand

- Erwartetes Ereignis: Breakpoint wird entfernt.
- Status: **FEHLSCHLAG**
Breakpoint kann nicht mehr entfernt werden, wenn die Zeile so modifiziert wurde, dass sie keinen Statement mehr enthält

1.3.2 Globalbreakpointview

1. Einfügen, Entfernen, Aktivieren, Deaktivieren von Globalbreakpoints im nicht-idle-Zustand

- Erwartetes Ereignis: Globalbreakpoints können nicht verändert werden, solange ein Programm läuft oder pausiert ist.
- Status: **OK**

2. Einfügen und Entfernen von syntaktisch korrekten Globalbreakpoints

- Erwartetes Ereignis: Der Breakpoint wird eingefügt bzw. entfernt.
- Status: **OK**

3. Einfügen von syntaktisch inkorrekten Globalbreakpoints

- Erwartetes Ereignis: Der Breakpoint wird nicht eingefügt.
- Status: **BEHOBEN**
Einfügen nach einem korrekt eingefügten Breakpoint bringt das Programm zum Absturz.

1.3.3 Helpbox

1. Es wird eine Stringkette eingegeben und nach Hilfe gesucht

- Erwartetes Ereignis: Der zur Stringkette am relevanteste Abschnitt wird in der Helpbox angezeigt.
- Status: **FEHLSCHLAG**
Wenn nach „else“ gesucht wird, erscheint die Einleitung

1.4 Buttons

1.4.1 CheckSyntax

1.4.2 Run

1.4.3 Single Step

1.4.4 Pause

1.4.5 Stop

1.4.6 Validate