



Bibliothek AutoGUITest V1.0

Windows-Benutzeroberflächen automatisiert testen

Tutorial

Ausgabe: 6.6.02



Inhalt

1 Übersicht	3
2 Funktionsweise	3
3 Funktionsumfang	3
4 Einsatz von AutoGUITest	4
4.1 Übersicht	4
4.2 TestApp: Die Software, die getestet werden soll	4
4.2.1 Aufbau	4
4.2.2 Was soll getestet werden ?	4
4.3 Erstellen der AutoGUITest-Applikation	5
4.3.1 Schritt 1: Erstellen des MFC-Projekts	5
4.3.2 Schritt 2: Schalter für Start des Tests platzieren	6
4.3.3 Schritt 3: Methode für den Start der AutoGUITest-Funktion erzeugen	7
4.3.4 Schritt 4: Einbinden von .h und .lib-Dateien	8
4.4 Erstellen des AutoGUITests	10
4.4.1 Allgemeines	10
4.4.2 Initialisierung von AutoGUITest	10
4.4.3 Pointer auf Applikation bestimmen	10
4.4.4 Resultat des Tests auswerten	11
4.4.5 Dialog oder Fenster über Menüpunkt öffnen	12
4.4.6 Wert in Eingabefeld eines Dialogs schreiben	13
4.4.7 Schalter betätigen	14
4.4.8 Meldungsfenster prüfen	15
4.5 Durchführen des AutoGUITests	16



1 Übersicht

Mit der Bibliothek **AutoGUITest** können unter Visual C++ einfach automatisierte Tests für Benutzeroberflächen von Windows-Applikationen implementiert werden. Besonders ideal dürfte die Bibliothek für Entwickler sein, welche Visual C++ bereits für die Implementation von Windows-Applikationen einsetzen. Sie können die automatisierten Tests in derselben Umgebung (Visual Studio) entwickeln, mit der sie täglich arbeiten.

AutoGUITest ist als herkömmliche DLL realisiert und verfügt deshalb nach aussen über normale Funktionsaufrufe.

2 Funktionsweise

- Die Funktionen der Bibliothek **AutoGUITest** erlauben den Zugriff auf die Fenster, Dialoge, Eingabefelder, Schalter, Menus und Meldungstexte einer beliebigen Windows-Applikation.
- Für den automatisierten Test wird mit Visual C++ eine Applikation entwickelt, welche über die Funktionen von **AutoGUITest** die zu testende Applikation fernsteuert.
- Der automatische Test wird Schritt für Schritt durchgeführt, bis das Ende des Tests erreicht ist oder bis ein Fehler auftritt. Fehler bedeutet in diesem Fall, dass sich die zu testende Software nicht so verhalten hat, wie das im Test erwartet wurde. Wenn ein Fehler auftritt, werden die Nummer des Testschritts und der Fehlercode gespeichert.

3 Funktionsumfang

Die Version 1.0 von **AutoGUITest** unterstützt Funktion für den Test von Benutzeroberflächen mit folgenden Eingabefelder, Schalter und Menus:

- Statisches Textfeld (CStatic).
- Textfeld (CEdit)
- Menu (CMenu)
- Schalter (CButton)
- Kontrollkästchen (CheckBox, CButton)
- Auswahlfeld (CComboBox)
- Auswahlliste (CListBox)

Die verfügbaren Funktionen sind in der Online-Referenz ausführlich beschrieben.

Die Bibliothek wird laufend erweitert.

4 Einsatz von AutoGUITest

4.1 Übersicht

Im folgenden Beispiel werden einige Funktionen von **AutoGUITest** in einer MFC-Applikation (Dialogfeldbasierend) eingebunden. Getestet werden soll ein einfacher Dialog einer SDI-Applikation. Das Beispiel wird in Visual C++ 6.0 realisiert.

4.2 TestApp: Die Software, die getestet werden soll

4.2.1 Aufbau

- Die Software *TestApp* ist eine SDI-Applikation, welche über den Dialog *Testdialog* verfügt.



- Der Dialog *Testdialog* wird über den Menüpunkt *Test / Dialog* (ID = 32771) geöffnet.
- Der Dialog enthält ein einziges Eingabefeld (CEdit, ID = 1000), in welches eine Zeichenkette erfasst werden muss, bevor der Dialog über den Schalter *OK* geschlossen werden kann.
- Wenn kein Wert im Eingabefeld erfasst und der Schalter *OK* betätigt wird, so wird eine Fehlermeldung (Titel: *Information*; Meldungstexte: *Eine Eingabe ist erforderlich !*) angezeigt.

4.2.2 Was soll getestet werden ?

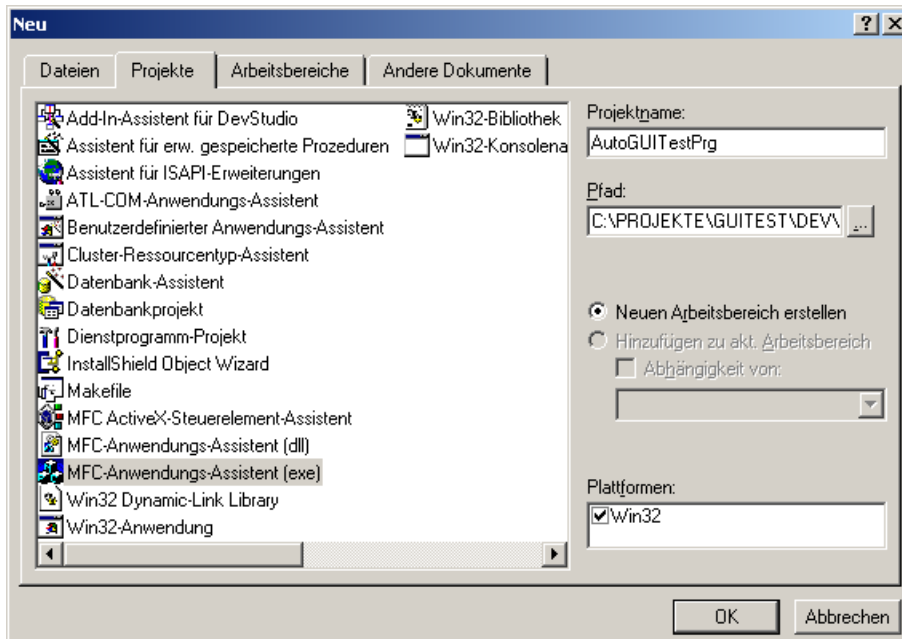
Der **AutoGUITest** soll in den folgenden Schritten ablaufen:

1. Testen, ob *TestApp* gestartet wurde.
2. Über den Menüpunkt *Test / Dialog* den Dialog *Testdialog* öffnen.
3. Text in Eingabefeld eingeben. Prüfen ob der Text in das Eingabefeld geschrieben wurde.
4. Schalter *OK* betätigen.
5. Dialog *Testdialog* nochmals über das Menu öffnen.
6. Kein Text in Eingabefeld schreiben und Schalter *OK* betätigen.
7. Fehlermeldung prüfen, anschliessend Meldungsfenster über Schalter schliessen.
8. Resultat des AutoGUITests ausgeben.

4.3 Erstellen der AutoGUITest-Applikation

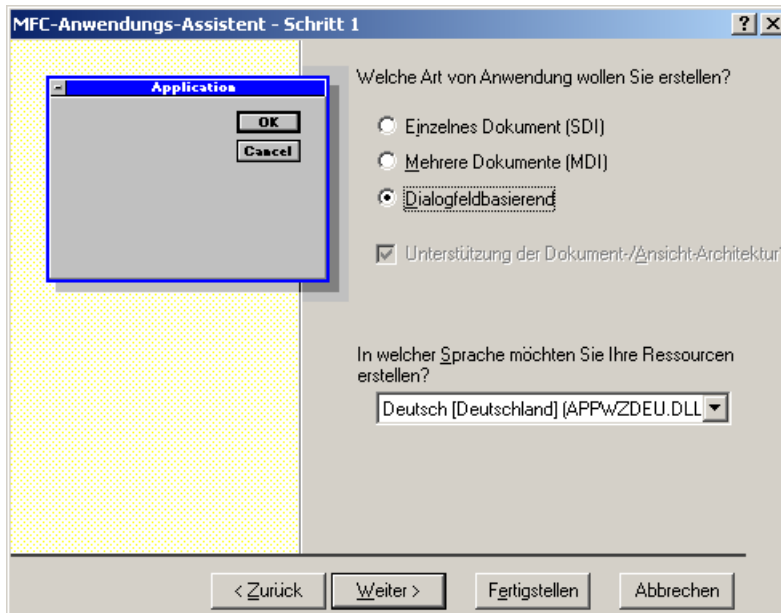
4.3.1 Schritt 1: Erstellen des MFC-Projekts

Das MFC-Projekt wird mit dem *MFC-Anwendungs-Assistenten (.exe)* erstellt.
Nach der Wahl des Menüpunkts *Datei / Neu* wird der Dialog *Neu* angezeigt.



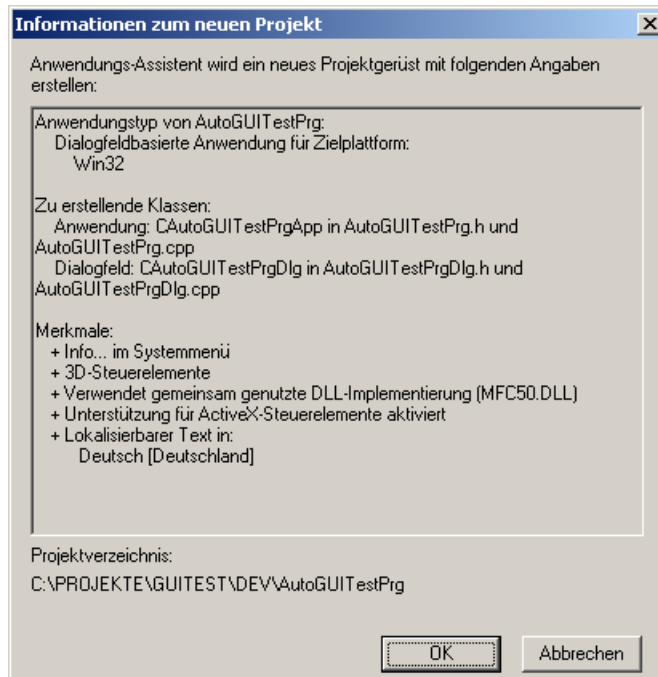
Nach der Definition des Projektnamens und der Wahl eines Pfades kann der Schalter *OK* betätigt werden.

Der Dialog *MFC-Anwendungs-Assistent - Schritt 1* wird angezeigt.



Nach der Wahl des Punktes *Dialogfeldbasierend* kann der Schalter *Fertigstellen* betätigt werden.

Der Dialog *Informationen zum neuen Projekt* wird angezeigt.

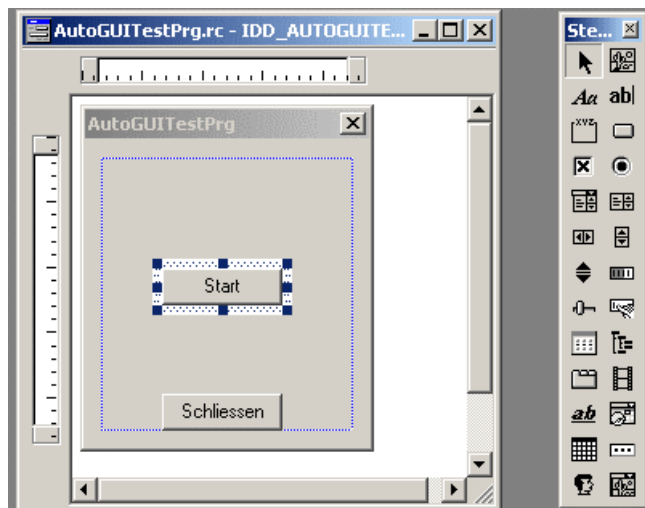


Nach der Betätigung des Schalters *OK* wird das Projekt erzeugt.

4.3.2 Schritt 2: Schalter für Start des Tests platzieren

Der **AutoGUITest** soll über den Schalter *Start* gestartet werden.

Der Schalter wird mit dem Ressourceneditor auf dem Dialog *CAutoGUITestPrgDlg* platziert.

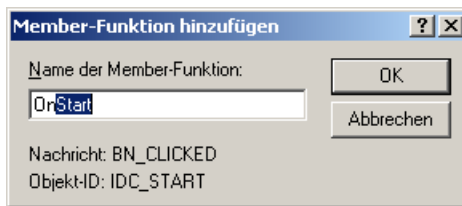


Im Dialog *Schaltfläche Eigenschaften* wird die ID des Schalters auf *IDC_START* gesetzt.



4.3.3 Schritt 3: Methode für den Start der AutoGUITest-Funktion erzeugen

Mit dem Klassen-Assistent wird die Member-Funktion *OnStart* (Message BN_CLICKED) erzeugt.



Erzeugt wird der untenstehende Quellcode.

```
AutoGUITestPrgDlg.cpp
// Die Systemaufrufe fragen den Cursorform ab, die angezeigt we
// das zum Symbol verkleinerte Fenster mit der Maus zieht.
HCURSOR CAutoGUITestPrgDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

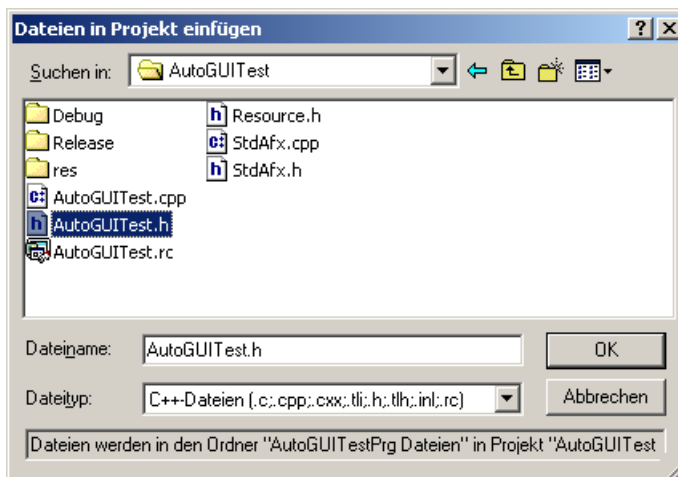
-----
// Test starten
-----
void CAutoGUITestPrgDlg::OnStart()
{
    // Deklarationen
}
-----
```

4.3.4 Schritt 4: Einbinden von .h und .lib-Dateien

Die Datei *AutoGUITest.h* muss zum Projekt hinzugefügt werden. Im Fenster Arbeitsbereich wird zu diesem Zweck der Eintrag *AutoGUITestPrg Dateien* markiert und anschliessend die rechte Maustaste betätigt.

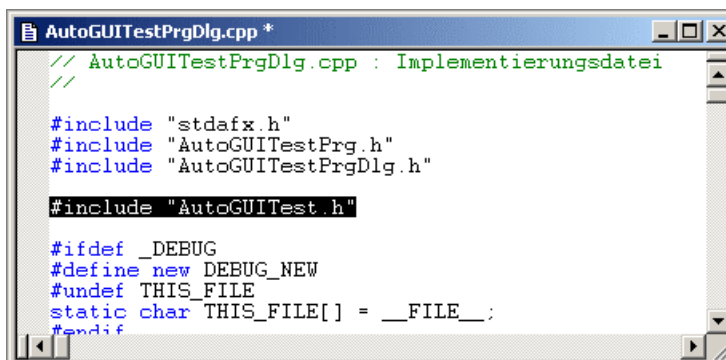


Nach der Wahl des Menüpunktes *Dateien zu Projekt hinzufügen* wird der Dialog *Dateien in Projekt einfügen* geöffnet. In diesem Dialog kann die Datei *AutoGUITest.h* ausgewählt werden.

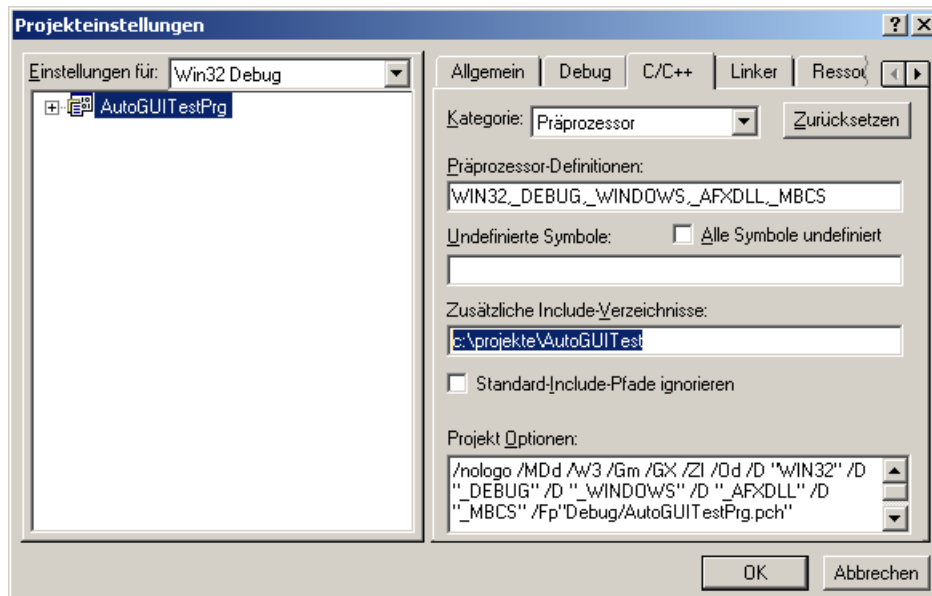


Nach der Betätigung des Schalters *OK* wird die Datei *AutoGUITest.h* in das Projekt eingefügt.

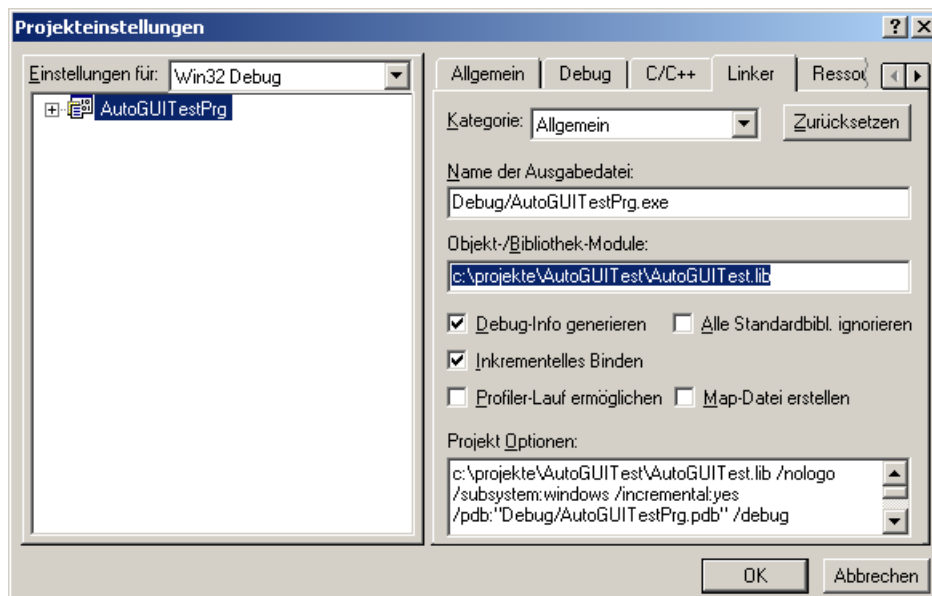
Die Datei *AutoTestGUI.h* wird mit einem *#include* in die Datei *AutoGUITestPrgDlg.cpp* eingebunden.



Damit der Compiler die Datei *AutoGUITest.h* finden kann, muss in den Projekteinstellungen im Tab *C/C++* in der Kategorie *Präprozessor* im Eingabefeld *Zusätzliche Include-Verzeichnisse* der Pfad auf das Verzeichnis erfasst werden, in welchem sich die Datei *AutoGUITest.h* befindet.



Bevor der eigentliche **AutoGUITest** implementiert werden kann, muss nun nur noch die *.lib*-Datei eingebunden werden. In den Projekteinstellungen muss im Tab *Linker* in der Kategorie *Allgemein* im Eingabefeld *Objekt-/Bibliothek-Module* der vollständige Name (inkl. Laufwerk und Pfad) der Datei *AutoGUITest.lib* erfasst werden.





4.4 Erstellen des AutoGUITests

4.4.1 Allgemeines

Bei der Erstellung eines **AutoGUITests** müssen folgende Punkte beachtet werden:

- **AutoGUITest** speichert den Status des Tests in einer eigenen Statusvariable. Wenn ein Fehler auftritt, werden alle nachfolgenden Testschritte nicht ausgeführt. Somit genügt es im Prinzip, wenn der Status am Ende des **AutoGUITests** abgefragt wird.
- Jedem Testschritt wird eine eindeutige Nummer zugewiesen. Mit Hilfe dieser Nummer kann der Testschritt bestimmt werden, beim dem sich die zu testende Applikation nicht wie erwartet verhalten hat.

4.4.2 Initialisierung von AutoGUITest

Die Initialisierung eines **AutoGUITests** erfolgt durch den Aufruf der Funktion *AgtInit()*.

Im Quellcode:

```
//-----  
// Test starten  
//-----  
void CAutoGUITestPrgDlg::OnStart()  
{  
    // AutoGUITest initialisieren  
    AgtInit(this);  
}  
//-----
```

4.4.3 Pointer auf Applikation bestimmen

Als erster Schritt eines **AutoGUITest** muss immer der Pointer auf die zu testende Applikation bestimmte werden. Die Funktion *AgtGetWndApp(...)* liefert den Pointer auf die Applikation.

Im Quellcode:

```
//-----  
// Test starten  
//-----  
void CAutoGUITestPrgDlg::OnStart()  
{  
    // Deklarationen  
    CWnd*      poWndApp = NULL;  
  
    // AutoGUITest initialisieren  
    AgtInit(this);  
  
    // Pointer auf Applikation  
    poWndApp = AgtGetWndApp(1, "TestApp");  
    if (poWndApp != NULL)  
    {  
        // Hier beginnt der eigentliche Test  
  
    }  
    else  
    {  
        // Zu testende Applikation muss gestartet werden  
        this->MessageBox("Starten Sie bitte TestApp !");  
    }  
}  
//-----
```

Bemerkungen:

- Der Pointer auf eine Applikation wird über den in der Titelzeile der Applikation angezeigten Namen ermittelt („TestApp“ im Quellcode).



4.4.4 Resultat des Tests auswerten

Das Resultat eines **AutoGUITests** wird als letzte Aktion eines Tests durchgeführt. Wenn alle Testschritte fehlerfrei abgearbeitet wurden, wird die Anzahl der erfolgreich durchgeführten Testschritte am Bildschirm ausgegeben. Im Fehlerfall wird die Nummer des Testschritts ausgegeben, in dem der Fehler aufgetreten ist. Zusätzlich wird der Code des Fehlers angezeigt.

Im Quellcode:

```
//-----  
// Test starten  
//-----  
void CAutoGUITestPrgDlg::OnStart()  
{  
    // Deklarationen  
    CWnd*      poWndApp = NULL;  
    CWnd*      poDlg = NULL;  
    CString    oMsg;  
  
    // AutoGUITest initialisieren  
    AgtInit(this);  
  
    // Pointer auf Applikation  
    poWndApp = AgtGetWndApp(1, "TestApp");  
    if (poWndApp != NULL)  
    {  
  
        // Ergebnis des AutoGUITests auswerten  
        if (AgtGetState() != AGT_STATE_OK)  
        {  
            oMsg.Format("Fehler %d in Testschritt %d !",  
                        AgtGetState(), AgtGetTestNr());  
            this->MessageBox(oMsg, "Information", MB_ICONINFORMATION);  
        }  
        else  
        {  
            oMsg.Format("Erfolgreich ausgeführte Testschritte: %d",  
                        AgtGetTestStepCnt());  
            this->MessageBox(oMsg, "Information", MB_ICONINFORMATION);  
        }  
    }  
    else  
    {  
        // Zu testende Applikation muss gestartet werden  
        this->MessageBox("Starten Sie bitte die Applikation TestApp !");  
    }  
}  
//-----
```



4.4.5 Dialog oder Fenster über Menüpunkt öffnen

Für die Betätigung von Menüpunkten stehen folgende Funktionen zur Verfügung:

- *AgtMenuSelectAppSend()*
- *AgtMenuSelectAppPost()*
- *AgtMenuSelectWndSend()*
- *AgtMenuSelectWndPost()*

Die Funktionen *AgtMenuSelectApp...* werden für die Betätigung eines Menüpunktes auf Applikationsebene verwendet, während die Funktionen *AgtMenuSelectWnd...* für die Betätigung eines Menüpunktes aus einem Fenster (MDI oder SDI) oder einem Dialog verwendet werden.

Die Funktionen *AgtMenuSelect...Send* wird verwendet, wenn mit der Ausführung des nächsten AutoGUITests gewartet werden kann, bis die Operation vollständig ausgeführt ist. Anwendung, z.B. öffnen eines Fensters.

Die Funktionen *AgtMenuSelect...Post* wird verwendet, wenn mit der Ausführung des nächsten AutoGUITests **nicht** gewartet werden kann, bis die Operation vollständig ausgeführt ist. Anwendung, z.B. öffnen eines Dialogs.

Im Quellcode:

```
//-----  
// Test starten  
//-----  
void CAutoGUITestPrgDlg::OnStart()  
{  
    // Deklarationen  
    CWnd*      poWndApp = NULL;  
    CWnd*      poDlg = NULL;  
    CString    oMsg;  
  
    // AutoGUITest initialisieren  
    AgtInit(this);  
  
    // Pointer auf Applikation  
    poWndApp = AgtGetWndApp(1, "TestApp");  
    if (poWndApp != NULL)  
    {  
        // Dialog über Menu öffnen  
        AgtMenuSelectWndPost(2, poWndApp, ID_MENU_TEST_DIALOG);  
        poDlg = AgtGetDlg(3, "Testdialog");  
  
        // Ergebnis des AutoGUITests auswerten  
        if (AgtGetState() != AGT_STATE_OK)  
        {  
            oMsg.Format("Fehler %d in Testschritt %d !",  
                AgtGetState(), AgtGetTestNr());  
            this->MessageBox(oMsg, "Information", MB_ICONINFORMATION);  
        }  
        else  
        {  
            oMsg.Format("Erfolgreich ausgeführte Testschritte: %d",  
                AgtGetTestStepCnt());  
            this->MessageBox(oMsg, "Information", MB_ICONINFORMATION);  
        }  
    }  
    else  
    {  
        // Zu testende Applikation muss gestartet werden  
        this->MessageBox("Starten Sie bitte die Applikation TestApp !");  
    }  
}  
//-----
```

ID_MENU_TEST_DIALOG enthält die ID des Menüpunktes *Test / Dialog* des zu testenden Programms *TestApp*.



4.4.6 Wert in Eingabefeld eines Dialogs schreiben

Mit der Funktion *AgtTextSet* kann eine Zeichenkette in ein Eingabefeld (CEdit), in eine Textfeld (CStatic) geschrieben oder der Titel eines Fensters/Dialogs gesetzt werden.

Mit der Funktion *AgtTextGet* kann eine Zeichenkette aus einem Eingabefeld (CEdit), aus einem Textfeld (CStatic) ausgelesen oder der Titel eines Fensters/Dialogs bestimmt werden.

Mit den Funktionen

- *AgtTextCheck*,
- *AgtTextCheckBool* und
- *AgtTextEmpty*

kann geprüft werden, ob ein Eingabefelde (CEdit) oder eine Textfeld (CStatic) die erwartete Zeichenkette enthält oder leer ist.

Im Quellcode:

```
//-----  
// Test starten  
//-----  
void CAutoGUITestPrgDlg::OnStart()  
{  
    // Deklarationen  
    CWnd*      poWndApp = NULL;  
    CWnd*      poDlg = NULL;  
    CString    oMsg;  
    CString    oTmp;  
  
    // AutoGUITest initialisieren  
    AgtInit(this);  
  
    // Pointer auf Applikation  
    poWndApp = AgtGetWndApp(1, "TestApp");  
    if (poWndApp != NULL)  
    {  
        // Dialog über Menu öffnen  
        AgtMenuSelectWndPost(2, poWndApp, ID_MENU_TEST_DIALOG);  
        poDlg = AgtGetDlg(3, "Testdialog");  
  
        // Text "Das ist ein Test" in Textfeld schreiben und  
        // testen, ob das Feld den Wert wirklich enthält.  
        oTmp = "Das ist ein Test";  
        AgtTextSet(4, poDlg, IDC_TD_EDIT, oTmp);  
        AgtTextCheck(5, poDlg, IDC_TD_EDIT, oTmp);  
  
        // Ergebnis des AutoGUITests auswerten  
        if (AgtGetState() != AGT_STATE_OK)  
        {  
            oMsg.Format("Fehler %d in Testschritt %d !",  
                AgtGetState(), AgtGetTestNr());  
            this->MessageBox(oMsg, "Information", MB_ICONINFORMATION);  
        }  
        else  
        {  
            oMsg.Format("Erfolgreich ausgeführte Testschritte: %d",  
                AgtGetTestStepCnt());  
            this->MessageBox(oMsg, "Information", MB_ICONINFORMATION);  
        }  
    }  
    else  
    {  
        // Zu testende Applikation muss gestartet werden  
        this->MessageBox("Starten Sie bitte die Applikation TestApp !");  
    }  
}  
//-----
```

ID_TD_EDIT enthält die ID des Eingabefeldes, in das der Wert geschrieben werden soll. Das Eingabefeld befindet sich im Dialog *Testdialog* der zu testenden Software *TestApp*.



4.4.7 Schalter betätigen

Für die Betätigung eines Schalters in einem Fenster oder einem Dialog stehen die Funktionen

- `AgtButtonSelectPost` und
- `AgtButtonSelectSend`

zur Verfügung.

`AgtButtonSelectPost` wird verwendet, wenn im Testprogramm die Ausführung nach der Betätigung des Schalters sofort fortgesetzt werden soll. Diese Funktion muss verwendet werden, wenn z.B. über einen Schalter ein Dialog geöffnet wird oder wenn über einen Schalter eine Plausibilitätsprüfung gestartet wird, welche Fehlermeldungen generieren kann.

`AgtButtonSelectSend` wird verwendet, wenn im Testprogramm die Ausführung nach der Betätigung des Schalters erst fortgesetzt werden soll, wenn die Funktion im zu testenden Programm abgeschlossen ist.

Im Quellcode:

```
//-----  
// Test starten  
//-----  
void CAutoGUITestPrgDlg::OnStart()  
{  
    // Deklarationen  
    CWnd*      poWndApp = NULL;  
    CWnd*      poDlg = NULL;  
    CString    oMsg;  
    CString    oTmp;  
  
    // AutoGUITest initialisieren  
    AgtInit(this);  
  
    // Pointer auf Applikation  
    poWndApp = AgtGetWndApp(1, "TestApp");  
    if (poWndApp != NULL)  
    {  
        // Dialog über Menu öffnen  
        AgtMenuSelectWndPost(2, poWndApp, ID_MENU_TEST_DIALOG);  
        poDlg = AgtGetDlg(3, "Testdialog");  
  
        // Text "Das ist ein Test" in Textfeld schreiben und  
        // testen, ob das Feld den Wert wirklich enthält.  
        oTmp = "Das ist ein Test";  
        AgtTextSet(4, poDlg, IDC_TD_EDIT, oTmp);  
        AgtTextCheck(5, poDlg, IDC_TD_EDIT, oTmp);  
  
        // Schalter OK betätigen; Dialog wird geschlossen  
        AgtButtonSelectPost(6, poDlg, IDOK);  
  
        // Ergebnis des AutoGUITests auswerten  
        if (AgtGetState() != AGT_STATE_OK)  
        {  
            oMsg.Format("Fehler %d in Testschritt %d !",  
                        AgtGetState(), AgtGetTestNr());  
            this->MessageBox(oMsg, "Information", MB_ICONINFORMATION);  
        }  
        else  
        {  
            oMsg.Format("Erfolgreich ausgeführte Testschritte: %d",  
                        AgtGetTestStepCnt());  
            this->MessageBox(oMsg, "Information", MB_ICONINFORMATION);  
        }  
    }  
    else  
    {  
        // Zu testende Applikation muss gestartet werden  
        this->MessageBox("Starten Sie bitte die Applikation TestApp !");  
    }  
}
```



4.4.8 Meldungsfenster prüfen

Wenn in der Software *TestApp* im Dialog *Testdialog* keine Eingabe in das Feld Eingabefeld vorgenommen wurde und der Schalter *OK* betätigt wird, wird die Fehlermeldung *Eine Eingabe ist erforderlich !* angezeigt.

Mit der Funktion *AgtMsgCheck* (oder *AgtMsgCheckBool*) wird geprüft, ob eine Meldung mit dem richtigen Meldungstext angezeigt wird. Weiter wird mit dieser Funktion festgelegt, über welchen Schalter das Meldungsfenster geschlossen werden soll.

Im Quellcode:

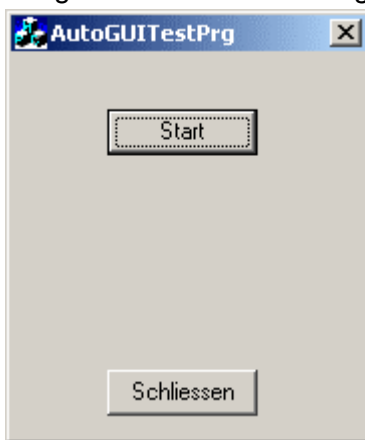
```
//-----  
// Test starten  
//-----  
void CAutoGUITestPrgDlg::OnStart()  
{  
    // Deklarationen  
    CWnd*      poWndApp = NULL;  
    CWnd*      poDlg = NULL;  
    CString    oMsg;  
    CString    oTmp;  
  
    // AutoGUITest initialisieren  
    AgtInit(this);  
  
    // Pointer auf Applikation  
    poWndApp = AgtGetWndApp(1, "TestApp");  
    if (poWndApp != NULL)  
    {  
        // Dialog über Menu öffnen  
        AgtMenuSelectWndPost(2, poWndApp, ID_MENU_TEST_DIALOG);  
        poDlg = AgtGetDlg(3, "Testdialog");  
  
        // Text "Das ist ein Test" in Textfeld schreiben und  
        // testen, ob das Feld den Wert wirklich enthält.  
        oTmp = "Das ist ein Test";  
        AgtTextSet(4, poDlg, IDC_TD_EDIT, oTmp);  
        AgtTextCheck(5, poDlg, IDC_TD_EDIT, oTmp);  
  
        // Schalter OK betätigen; Dialog wird geschlossen  
        AgtButtonSelectPost(6, poDlg, IDOK);  
  
        // Dialog nochmals über Menu öffnen  
        AgtMenuSelectWndPost(7, poWndApp, ID_MENU_TEST_DIALOG);  
        poDlg = AgtGetDlg(8, "Testdialog");  
  
        // Schalter OK betätigen  
        // - Fehlermeldung muss angezeigt werden,  
        // - Dialog bleibt offen  
        AgtButtonSelectPost(9, poDlg, IDOK);  
  
        // Testen, ob Meldung ausgegeben wird  
        AgtMsgCheck(10, "Information", "Eine Eingabe ist erforderlich !", IDCANCEL);  
  
        // Text in Eingabefeld schreiben  
        oTmp = "Noch ein Test";  
        AgtTextSet(11, poDlg, IDC_TD_EDIT, oTmp);  
  
        // Schalter OK betätigen  
        // - Dialog wird geschlossen  
        AgtButtonSelectPost(9, poDlg, IDOK);  
  
        // Ergebnis des AutoGUITests auswerten  
        if (AgtGetState() != AGT_STATE_OK)  
        {  
            oMsg.Format("Fehler %d in Testschritt %d !",  
                        AgtGetState(), AgtGetTestNr());  
            this->MessageBox(oMsg, "Information", MB_ICONINFORMATION);  
        }  
        else  
        {  
            oMsg.Format("Erfolgreiche Testschritte: %d", AgtGetTestStepCnt());  
            this->MessageBox(oMsg, "Information", MB_ICONINFORMATION);  
        }  
    }  
}
```

```
    }  
  }  
  else  
  {  
    // Zu testende Applikation muss gestartet werden  
    this->MessageBox("Starten Sie bitte die Applikation TestApp !");  
  }  
}  
//-----
```

4.5 Durchführen des AutoGUITests

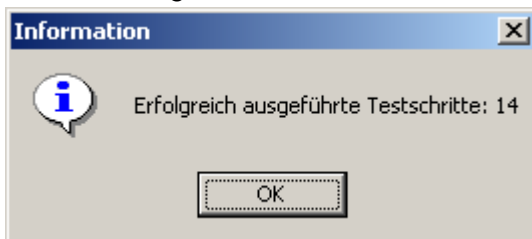
Nach dem fehlerfreien Kompilieren/Linken der AutoGUITest-Applikation erfolgt der Test in den folgenden Schritten:

1. Programm *TestApp* starten
2. Programm *AutoGUITestPrg* starten
3. Im Programm *AutoGUITestPrg* den Schalter *Start* betätigen

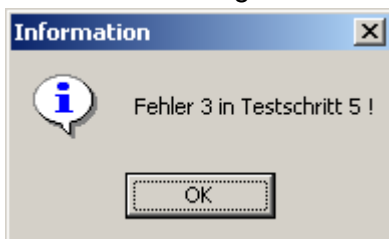


Der Test wird gestartet und durchgeführt.

4. Nach dem erfolgreichen Abschluss des Test wird die folgende Meldung angezeigt:



Im Fehlerfall wird folgende Meldung angezeigt:



In diesem Beispiel enthielt in Testschritt 5 das Eingabefeld nicht den erwarteten Text.