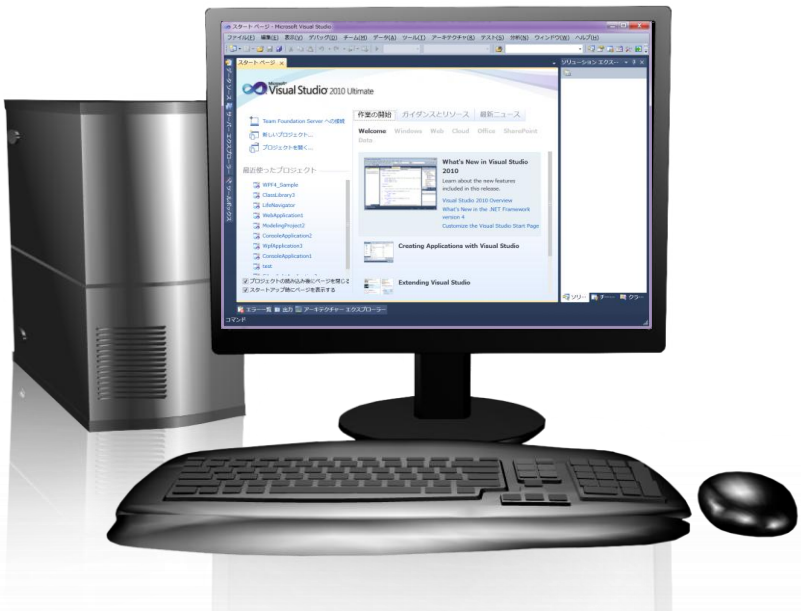
Microsoft® Visual Studio® 2010

Microsoft® Visual Studio® 2010 評価ガイド

～ ユーザーインターフェイスの自動テスト ～



著作権

このドキュメントに記載されている情報は、このドキュメントの発行時点におけるマイクロソフトの見解を反映したものです。マイクロソフトは市場の変化に対応する必要があるため、このドキュメントの内容に関する責任を問われないものとします。また、発行日以降に発表される情報の正確性を保証できません。
このホワイトペーパーは情報提供のみを目的としています。明示、黙示、または法令に基づく規定に関わらず、これらの情報についてマイクロソフトはいかなる責任も負わないものとします。
この文書およびソフトウェアを使用する場合は、適用されるすべての著作権関連の法律に従っていただくものとします。このドキュメントのいかなる部分も、米国 Microsoft Corporation の書面による許諾を受けることなく、その目的を問わず、どのような形態であっても、複製または譲渡することは禁じられています。ここでいう形態とは、複写や記録など、電子的な、または物理的なすべての手段を含みます。ただしこれは、著作権法上のお客様の権利を制限するものではありません。
マイクロソフトは、この文書に記載されている事項に関して、特許、申請中特許、商標、著作権、および他の知的財産権を所有する場合があります。別途マイクロソフトのライセンス契約上に明示の規定のない限り、このドキュメントはこれらの特許、商標、著作権、またはその他の知的財産権に関する権利をお客様に許諾するものではありません。
別途記載されていない場合、このドキュメントで使用している会社、組織、製品、ドメイン名、電子メール アドレス、ロゴ、人物、場所、出来事などの名称は架空のものです。実在する商品名、団体名、個人名などとは一切関係ありません。

目次

ユーザーインターフェイスの自動テスト

アプリケーションのユーザーインターフェイス (UI) の動作確認は、人による操作が必要となるため工数がかかります。テストの効率化のためには、UI の動作確認テストについて自動化できることが望ましいです。Visual Studio 2010 では「コード化された UI テスト」機能によって、UI テストの自動化を支援します。

コード化された UI テストの実施には、Visual Studio 2010 Ultimate または Visual Studio 2010 Premium が必要です。これら製品の評価版は以下のサイトより入手いただけます。
http://www.microsoft.com/japan/visualstudio/download

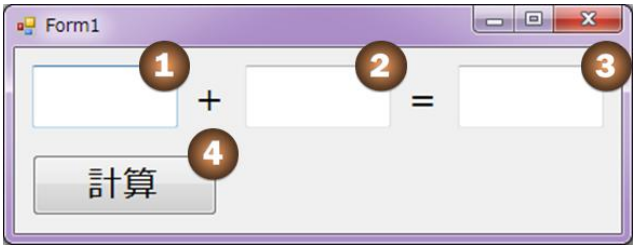
[事前準備] UI 自動テストの対象のプログラムの作成

本ドキュメントで使用するテスト対象プログラムを作成します。
ここでは Windows フォーム アプリケーションを作成します。言語は C# でも Visual Basic でも構いません。
この手順に従って作成しても、既にお持ちのアプリケーションを使用しても構いません。

■ UI の作成

以下のコントロールをフォームに貼り付け、それぞれプロパティを定義します。
基本的に 3 つのテキストボックスとボタンを貼り付けただけです。以下の例では、わかりやすく説明するために、フォントサイズを大きくしたり、”+”, “=” というラベル コントロールを貼り付けていますが、本ドキュメントの動作上問題ないので、割愛して構いません。

番号	コントロール	プロパティ	値
①	TextBox (足し算のパラメーター 1 を入力)	(Name)	VB: TextBox1 C#: textBox1 ※デフォルトのまま
②	TextBox (足し算のパラメーター 2 を入力)	(Name)	VB: TextBox2 C#: textBox2 ※デフォルトのまま
③	TextBox (足し算の結果を表示)	(Name)	VB: TextBox3 C#: textBox3 ※デフォルトのまま
④	Button (足し算を実行)	(Name)	VB: Button1 C#: button1 ※デフォルトのまま



ロジックの作成

“計算” ボタンがクリックされたときに、足し算を行うプログラムを記述します。

【C#】

```
private void button1_Click(object sender, EventArgs e)
{
    int x;
    int y;
    int answer;

    x = Int32.Parse(textBox1.Text);
    y = Int32.Parse(textBox2.Text);
    answer = x + y;

    textBox3.Text = answer.ToString();
}
```

【VB】

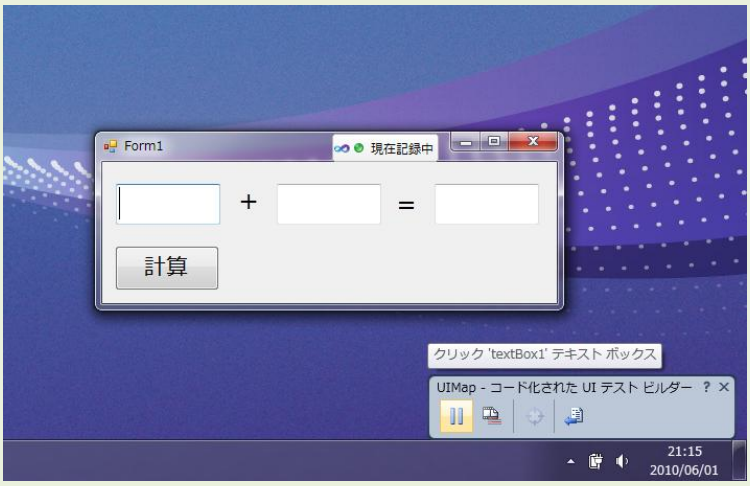
```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim x As Integer
    Dim y As Integer
    Dim answer As Integer

    x = Int32.Parse(TextBox1.Text)
    y = Int32.Parse(TextBox2.Text)
    answer = x + y

    TextBox3.Text = answer.ToString()
End Sub
```

基本的な UI 自動テストの実施

UI の操作を記録し、それを単純に再生してみましょう。

■ UI 自動テストの作成

1. アプリケーションの UI テストを行う準備をします。(例: あらかじめアプリケーションを起動するなど)
2. Visual Studio 2010 を起動します。
3. メニューより [テスト] – [新しいテスト] を選択します。
4. [新しいテストの追加] ダイアログより、[コード化された UI テスト] を選択します。
5. [コード化された UI テストのコード生成] より “操作の記録、UI マップの編集、またはアサーションの追加” を選択し、[OK] ボタンをクリックします。
6. UI テストビルダーが起動します。一番左のオレンジ色の丸いボタン (記録の開始) をクリックし、テストを開始します。



7. テスト (記録する UI 操作) を実施します。操作が記録されているアプリケーションは “現在記録中” という文字がアプリケーションのタイトル バーに表示されます。
8. 記録された操作を確認する場合には、左から二番目のボタン (記録されたステップの表示) をクリックします。
9. 記録をいったん停止、またはテストを終了する場合は、一番左のボタン (操作の一時停止) をクリックします。
10. テストを完全に終了する場合、一番右のボタン (コードの生成) をクリックします。表示されるダイアログに、テスト メソッド名を入力し、[追加と生成] ボタンをクリックします。

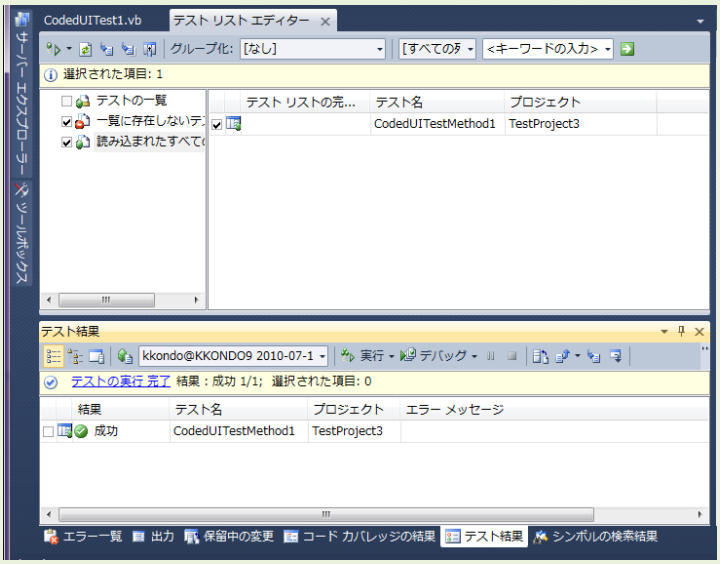
【Point】

記録された操作を削除する場合、 8 の手順で表示された [記録されたステップの表示] ダイアログから該当の操作を選択し、右クリックして、[選択した操作の削除] をクリックします。

作成した UI 自動テストを実行します。テストの実行方法は単体テストと同じです。詳細は以下の記事を参考にしてください。
http://msdn.microsoft.com/ja-jp/library/dd286656.aspx

■ UI 自動テストの実行

1. テストを再生する準備をします。(例: あらかじめアプリケーションを起動するなど)
2. Visual Studio 2010 を起動します。
3. メニューより [テスト] – [テスト リスト エディター] または [テスト ビュー] を選択します。
4. テストの一覧より、作成されたテスト (デフォルトでは “CodedUITestMethod1”) をチェック (または選択) し、テスト実行ボタンをクリックします。
   ※ テストが一覧に表示されない場合、テスト プロジェクトをリビルドしてみてください。
5. テストが実行されます。テストの実行結果は [テスト結果] ウィンドウに表示されます。
   ※ [テスト結果] ウィンドウが表示されない場合、メニューより [テスト] – [ウィンドウ] – [テスト結果] を選択ください。



コード化された UI テスト コードのファイルやクラス

■ 生成されるファイル

コード化された UI テストコードを作成すると、以下のファイルが生成されます。
中には隠しファイルとして作成されるものもあります。[ソリューション エクスプローラー] よりテスト プロジェクトを選択し、メニューより [プロジェクト] – [すべてのファイルを表示] をクリックしてください。

ファイル名	説明
CodedUITest1.vb または CodedUITest1.cs	テスト クラス、テスト メソッド、検証メソッド (アサーション) などが定義されています。
UIMap.uitest	テストで使用する UI コントロールの定義 (UI マップ) が書かれている XML ファイルで

	す。
UIMap.Designer.vb　または UIMap.Designer.cs	UIMap.uitest ファイルのコードが定義されています。 作成された 自動メソッド、検証メソッドの詳細はここから確認することができます。
UIMap.vb　または UIMap.cs	UIMap のクラス ファイルです。 UI マップのカスタマイズは、このファイルで行います。
UserControls.cs　または UserControls.vb	コード化された UI テストで使用される特殊クラスが定義されます。

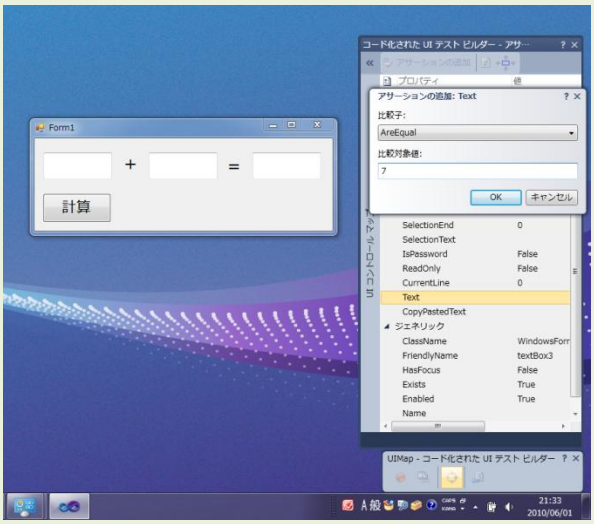
コード化された UI テストを複数追加しても、1 つの UI マップのみが作成されます。
したがって複数のテストを作成すると UI マップが大きくなります。そのため、画面単位や部品単位など、ある程度の単位でテスト プロジェクトを作成し、コード化された UI テストを分散することで、UI マップが大きくなるのを抑えます。

値の検証

UI 操作の記録をそのまま再生することで、操作のみの検証を行うこともできますが、場合によってはそのテストによってコントロールに正しく値が設定されているかを検証したい場合があります。
以下の手順で、特定のコントロールの値を検証することができます。

■ 値の検証

1. UI テストビルダーを起動します。【UI 自動テストの作成】の 1 ～ 5 のステップを実施してください。
   既存のテストをベースにする場合、該当のテスト メソッドを右クリックし、コンテキスト メニューより [コード化された UI テストのコードの生成] – [コード化された UI テスト ビルダーの使用] をクリックします。
2. 特定のコントロールの値を検証するには、UI テストビルダーの左から 3 番目の [アサーションの追加] ボタンを使用します。
   [アサーションの追加] ボタンをドラッグしながら該当のコントロール上でドロップします。
   (ドラック中に、青い四角が表示されますので、該当のコントロールが囲まれたところで、マウスのボタンを放してください)
3. プロパティの一覧が表示されますので、検証をしたいプロパティを選択し、[アサーションの追加] ボタンを押してください。
   (例: TextBox コントロールの入力値を確認する場合には、”Text” プロパティを選択)



4. 検証の式として [比較子] をドロップ ダウン リストから選択、検証対象の値として [比較対象値] に値を入力します。
5. 終了する場合、一番右のボタン (コードの生成) をクリックします。表示されるダイアログに、検証メソッド名を入力し、[追加と生成] ボタンをクリックします。
6. テスト メソッドに 5 で作成した検証メソッドが追加されていることを確認します。
   期待するメソッドに追加されていない場合、または独自に呼び出す場合などにおいては、以下のコードを記述します。

【C#】this.UIMap.AssertMethod1();
【VB】Me.UIMap.AssertMethod1()
※“AssertMethod1”は、5 で名前をつけた検証メソッド名

条件の変更

UI 操作の再現においてパラメーターを変更したい場合 (例えばテキスト ボックスの入力値を変更するなど)、生成されたテストコードを修正します。”UI 操作記録メソッド名 Params” (例: RecordedMethod1Params) というクラスが生成され、UI 操作における各パラメータに対して、それぞれプロパティが設定されます。
同様に上記【値の検証】の手順で作成した検証項目については、”検証メソッド名 ExptectedValues” (例: AssertMethod1ExpectedValues) というクラスが生成され、それぞれプロパティが設定されます。
条件を変更する場合には、UI 操作記録メソッド、および検証メソッドの呼び出し前に、それぞれプロパティを設定します。

1. 上記【値の検証】のステップを実行します。
2. ”UI 操作記録メソッド名 Params” クラスのプロパティを変更し、入力条件を変更します。
   また、”検証メソッド名 ExptectedValues” クラスのプロパティを変更し、期待する検証結果の条件を変更します。
   例えば、以下のようにコードを記述します。

[C#]

```
[TestMethod]
public void CodedUITestMethod1()
{
    this.UIMap.RecordedMethod1Params.UITextBox1EditText = “1”;
    this.UIMap.RecordedMethod1Params.UITextBox2EditText = “2”;
    this.UIMap.RecordedMethod1();

    this.UIMap.AssertMethod1ExpectedValues.UITextBox3EditText = “3”;
    this.UIMap.AssertMethod1();
}
```

[VB]

```
<TestMethod()>
Public Sub CodedUITestMethod1()
    Me.UIMap. RecordedMethod1Params.UITextBox1EditText = “1”
    Me.UIMap. RecordedMethod1Params.UITextBox1EditText = “2”
    Me.UIMap.RecordedMethod1()

    Me.UIMap.AssertMethod1ExpectedValues.UITextBox3EditText = “3”
    Me.UIMap.AssertMethod1()

End Sub
```

ひとつのテスト パターンを異なるデータで確認

例えば上限値、下限値といった “しきい値” の確認など、ひとつのテスト パターンを異なるデータで確認したいケースが考えられます。

上記の例では、固定値をテスト コードの各変数に直接設定しましたが、Visual Studio 2010 では単体テスト機能と同様、コード化された UI テストにおいても、テスト データを事前にデータベース、CSV 形式、XML 形式のいずれかの形式で定義し、各変数とデータ連結して動的に指定することで、テスト データの定義したテスト レコード数分、自動的にテストを実施させることができます。

1. テスト条件、および期待するテスト結果を含む CSV ファイルを作成します。
   例えば、2 つの値を加算するプログラムの確認であれば、x, y という 2 つの入力値と ExpectedResult という期待値を以下のように CSV ファイルに定義します。

```
x, y, ExpectedResult
1,2,3
2,4,6
10,10,20
```

2. 上記【値の検証】のステップを実行します。
3. [テスト リスト エディター] および [テスト ビュー] で該当のメソッドを選択します。
4. [プロパティ] ウィンドウより [データ接続文字列] プロパティに適切な接続文字列を設定します。
   プロパティの […] ボタンをクリックするとウィザードが起動され、1. で作成した CSV ファイルを指定します。
5. 必要に応じて [データ テーブル名] プロパティに該当のデータを含むテーブルを指定します)
6. [データ接続文字列] プロパティの設定により、テストメソッドに自動的に DataSource 属性が追加されていることが確認できます。

上記【条件の変更】と同じく、テストコードを変更しますが、TestContext.DataRow でテストデータを動的にバインドします。

[C#]

```
[DataSource (… 省略), TestMethod]
public void CodedUITestMethod1()
{
    this.UIMap.RecordedMethod1Params.UITextBox1EditText = (int) TestContext.DataRow["x"];
    this.UIMap.RecordedMethod1Params.UITextBox2EditText = (int) TestContext.DataRow["y"];
    this.UIMap.RecordedMethod1();

    this.UIMap.AssertMethod1ExpectedValues.UITextBox3EditText =
                                                (int) TestContext.DataRow["ExpectedResult"];
    this.UIMap.AssertMethod1();
}
```

[VB]

```
< DataSource (… 省略), TestMethod()>
Public Sub CodedUITestMethod1()
    Me.UIMap.RecordedMethod1Params.UITextBox1EditText = TestContext.DataRow(“x”)
    Me.UIMap.RecordedMethod1Params.UITextBox1EditText = TestContext.DataRow(“y”)
    Me.UIMap.RecordedMethod1()

    Me.UIMap.AssertMethod1ExpectedValues.UITextBox3EditText =
                                                TestContext.DataRow(“ExpectedResult”)
    Me.UIMap.AssertMethod1()
End Sub
```

7. 【基本的な UI 自動テストの実施】-【UI 自動テストの実行】の手順に従ってテストを実行します。
   CSV ファイルに定義したレコード数分テストが実施されていることを確認します。

独自のマウス、キーボードの操作

以下のクラスを利用することで、マウス、およびキーボードの UI 操作をプログラムすることができます。

クラス	解説
Mouse (Microsoft.VisualStudio.TestTools.UITesting)	マウスの操作を実行します。 Mouse.Click： クリック操作を実行します Mouse.DoubleClick： ダブルクリック操作を実行します
Keyboard (Microsoft.VisualStudio.TestTools.UITesting)	キーボードの操作を実行します。 Keyboard.SendKeys： キーボードの操作を実行します

【C#】

```
Mouse.Click (this.UIMap.UIForm1Window.UIButton1Window.UIButton1Button);
Keyboard.SendKeys (this.UIMap.UIForm1Window.UITextBox1Window.UITextBox1Edit,
                                                    “Kazuhiko{Space}Kondo{Enter}”);
```

【VB】

```
Mouse.Click (Me.UIMap.UIForm1Window.UIButton1Window.UIButton1Button)
Keyboard.SendKeys (Me.UIMap.UIForm1Window.UITextBox1Window.UITextBox1Edit,
                                                    “Kazuhiko{Space}Kondo{Enter}”)
```

※ コントロール名は UIMap ファイルに定義されています。

既存の UI 操作記録から UI テストを作成する

Microsoft® Test Manager 2010 を活用した手動テストにおいても、UI 操作の記録を取ることができます。Visual Studio 2010 では Test Manager 2010 が記録した UI 操作記録からコード化された UI テストを作成することができます。これはテスト担当者から報告された問題を修正する際に、開発者がその操作を繰り返しテストする必要がある場合に有用です。

以下の作業を確認するためには Test Manager 2010 で実施された UI 操作が必要です。Test Manager 2010 の操作方法については、別途ドキュメント等を参考にしてください。

■ 既存の UI 操作記録から UI テストを作成する

1. Test Manager 2010 の手動テストで記録された UI 操作を含む作業項目を用意します。
2. Visual Studio 2010 を起動します。
3. メニューより [テスト] – [新しいテスト] を選択します。
4. [新しいテストの追加] ダイアログより、[コード化された UI テスト] を選択します。
5. [コード化された UI テストのコード生成] より “既存の操作の記録を使用” を選択し、[OK] ボタンをクリックします。
6. [作業項目ピッカー] ダイアログが起動します。1. の作業項目を、クエリを活用して検索します。

クエリの活用については、Team Foundation Server の操作を参考ください。
該当の作業項目を選択し、[OK] ボタンをクリックします。

7. コード化された UI テストが作成されるのを確認します。
8. 値の検証 (アサーション) を追加するには、該当のテスト メソッドを右クリックし、[コード化された UI テストのコードの生成] – [コード化された UI テスト ビルダーの使用] をクリックします。
詳細は、上記【値の検証】を参照してください。

コード化された UI テストの要件

コード化された UI テストを実施できるシステム要件やアプリケーションの種類には制限があります。
現状は未サポートとなっているシステム要件やアプリケーションの一部については、現在サポートするための拡張機能を開発していますのでお待ちください。
詳細、および最新の情報については http://msdn.microsoft.com/ja-jp/library/dd380742.aspx を参照ください。

表: コード化された UI テストがサポートするシステム必須要件

オペレーティング システム	• Windows XP Service Pack 3 以降 • Windows Server 2003 Service Pack 1 以降 (*1) • Windows Server 2008 および Windows Server 2008 R2 (*1) • Windows Vista Service Pack 1 以降 • Windows 7 (*1) Windows Server 2003 および Windows Server 2008 を使用する場合は、Windows Internet Explorer セキュリティ強化をオフにする必要があります。
アーキテクチャ	x86 および x64 32 ビットの Windows では 32 ビットのアプリケーションをテストできます。 64 ビットの Windows で UI 同期を行う場合は、WOW 上で動作する 32 ビットのアプリケーションをテストできます。 64 ビットの Windows で UI 同期を行わない場合は、64 ビットの Windows フォームアプリケーション、または WPF アプリケーションをテストできます。 UI 同期とは、各コントロールのメッセージ キューで再生を検証する機能です。送信されたイベントに対してコントロールが応答しなかった場合は、イベントが再度送信されます。
.NET Framework のバージョン	• .NET Framework 2.0 • .NET Framework 3.0 • .NET Framework 3.5 • .NET Framework 4

	注意) Test Manager および Visual Studio 2010 が動作するためには .NET Framework 4 が必要です。
言語	英語、日本語、韓国語、ヒンディー語

表: コード化された UI テストがサポートするシステム要件とアプリケーションの種類の対応表

Internet Explorer 7.0 Internet Explorer 8.0 (* HTML, AJAX の動作含む)	サポート
FireFox 3.5 以上	以下の PowerTool を利用することで FireFox 上での自動 UI テストの再生が可能です。(操作の記録はできません) http://go.microsoft.com/fwlink/?LinkId=185264
Internet Explorer 6.0	未サポート
Google Chrome	未サポート
Opera	未サポート
Safari	未サポート
Silverlight	未サポート ※サポートするための機能を現在開発中
Windows フォーム アプリケーション	.NET Framework 2.0 以上をサポート (サードパーティ製品は除く)
WPF アプリケーション	.NET Framework 3.5 以上をサポート (サードパーティ製品は除く)
MFC アプリケーション	未サポート (動作する可能性がありますが、既知の問題がいくつか存在します)
Win32 アプリケーション	未サポート (動作する可能性がありますが、既知の問題がいくつか存在します)
Office アプリケーション (VSTO 含む)	未サポート

参考情報

[MSDN Library] 自動 UI テストを使用したユーザー インターフェイスのテスト

http://msdn.microsoft.com/ja-jp/library/dd286726.aspx

www.microsoft.com/japan/msdn/vstudio