機能拡張ツール開発解説書
目　　次

1. はじめに

本解説書では、機能拡張ツール（以降ツールと表現）のサンプルプロジェクトを例に、ツール開発者が対応する内容について説明します。本解説書に従い開発することで、CommonMPVer1.3で追加される下記の簡略化機能に対応することができます。

●ツール本体を含むファイル一式をパッケージ化
●パッケージ単位で CommonMP 本体へインストール及び削除
●起動用のランチャーからツールを起動

【補足事項】
・本解説書は、CommonMP Ver1.3 以降を対象としています。
・本解説書は、CommonMP 上で動作するツールを作成するツール開発者を対象としています。
・本解説書に従い開発したツールは、演算実行等のシステム側とは独立して動作する為、
ツール側から CommonMP の演算制御やＤＢ機能の使用等を行うことはできません。

2. ツール開発環境

マイクロソフト社の Visual Studio 上から、新しく開発するツールのデバッグが行える開発環境を¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥下に用意しました。

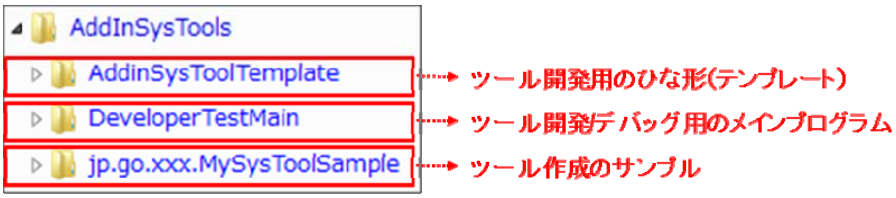

図 2.1　ツール開発環境とサンプルの提供

まず、¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥下の「SysToolTestMain.sln」を、Visual Studio から開きます。このソリューションに、ツール開発用のプロジェクトを追加してデバッグ機能を活用しながら開発を行います。サンプルとして、MySysToolSample.csproj が予め登録されています。以下、本サンプルに従って説明を行います。

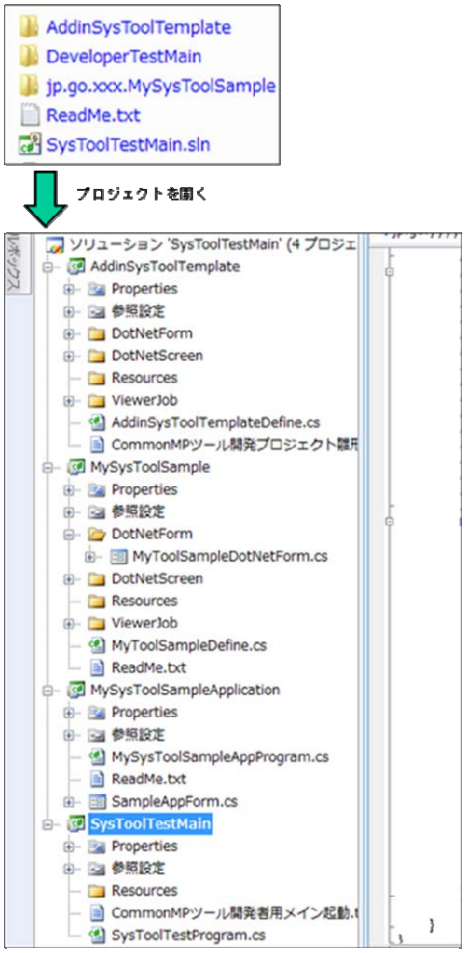

図 2.2　デバッグ用ソリューション立ち上げ

3.　サンプル機能の概要

まず、サンプルの機能説明を行います。

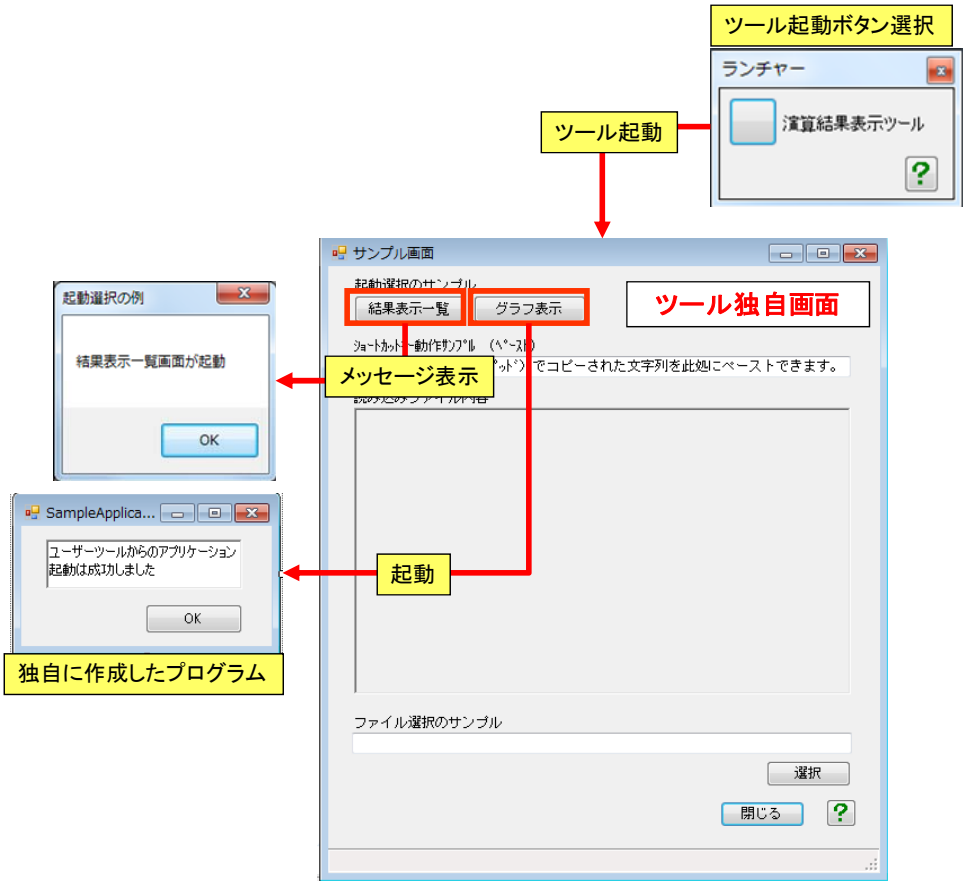

図 3.1　サンプル機能の概要図

図 3.1 に示すように、サンプルツールは Ver1.3 から追加されるランチャー画面から起動されます。本サンプルでは、ツール独自の画面を表示します。

この独自画面に配置されている「結果表示一覧」ボタンを押下するとメッセージが表示されます。また、「グラフ表示」ボタンを押下すると、サンプルプログラム（jp.go.xxx.MySysToolSampleApplication.exe）を起動し、そのプログラムが独自の画面を表示します。サンプルとしての機能は画面表示だけで、表示された画面の『OK』ボタンを押下すると、サンプルプログラムは終了します。

　本サンプルを開発する Visual Studio 用のプロジェクトは、CommonMP¥Source¥HYSSOP¥AddInSysTools¥jp.go.xxx.MySysToolSample 下に「MySysToolSample.csproj」として用意されています。

3.1　オペレータ操作とツール起動に至る処理概要

ツール開発に当たっては、ランチャー画面からどのような処理でツールが起動されるかを、理解しておくことが必要です。図 3.2 に利用者がランチャー画面を選択してから、ツールが起動されるまでの処理概要を示します。

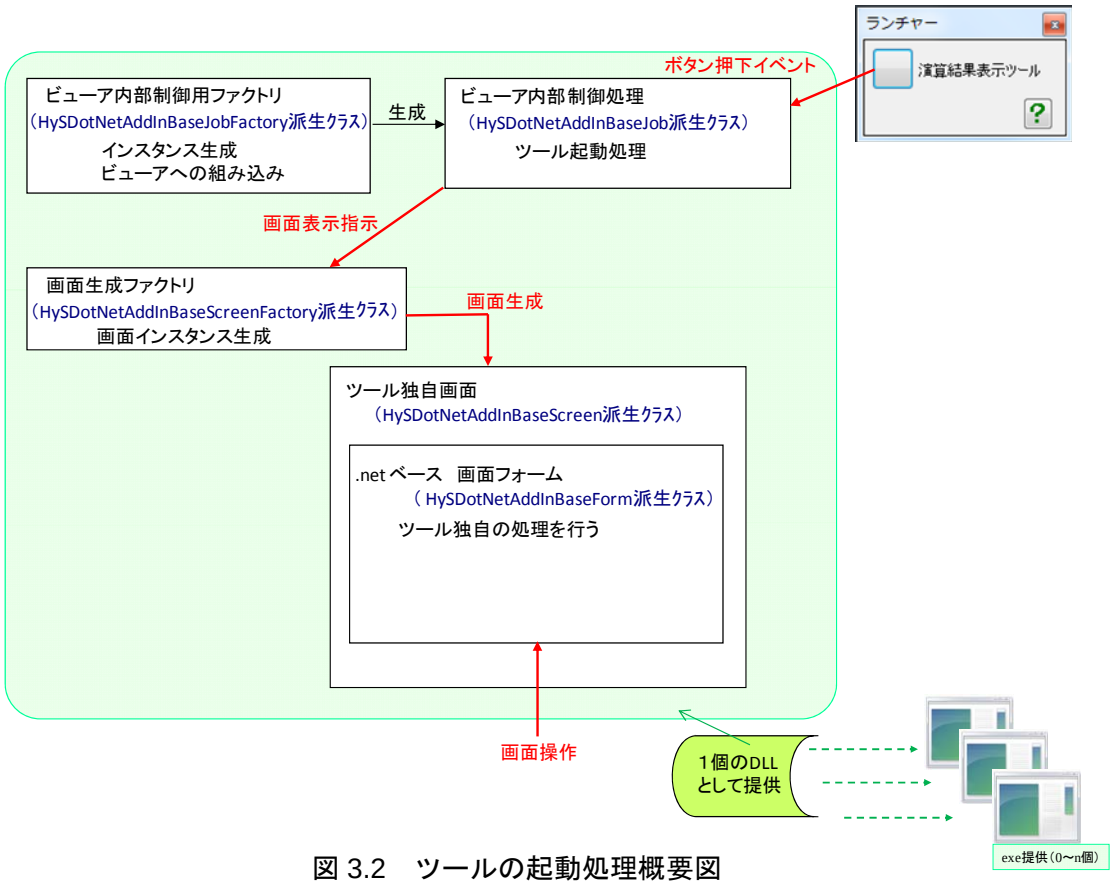

図 3.2　ツールの起動処理概要図

利用者がランチャー画面のボタンを押下すると、押下したイベントがビューア内部制御（業務）処理に通知されます。ビューア内部制御処理は、ツール開発者の実装内容に従ってツール起動処理を開始します。本サンプルのように画面表示要求であれば、画面生成ファクトリに独自画面を生成させ、画面を表示させます。図 3.2 に示す以下のクラスは、一つの DLL として提供される必要があります。

・ビューア内部制御用ファクトリ
・ビューア内部制御処理
・画面生成ファクトリ
・ツール独自画面
・.net ベース画面フォーム

ただし、本サンプルのように、ツール(DLL)から開発者が作成したアプリケーション(exe)を起動することは可能です。

3.2　ツールの親クラスとツール開発者が作成するクラス

CommonMP で提供されるツールの親クラスとツール開発者が作成する各クラスの派生関係を図 3.3 に示します（図の右側、派生して作成するクラス名は本サンプルを例としています)。

なお、サンプルのソースは　CommonMP をインストールしたディレクトリ下の¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥jp.go.xxx.MySysToolSample¥ 下に存在します。

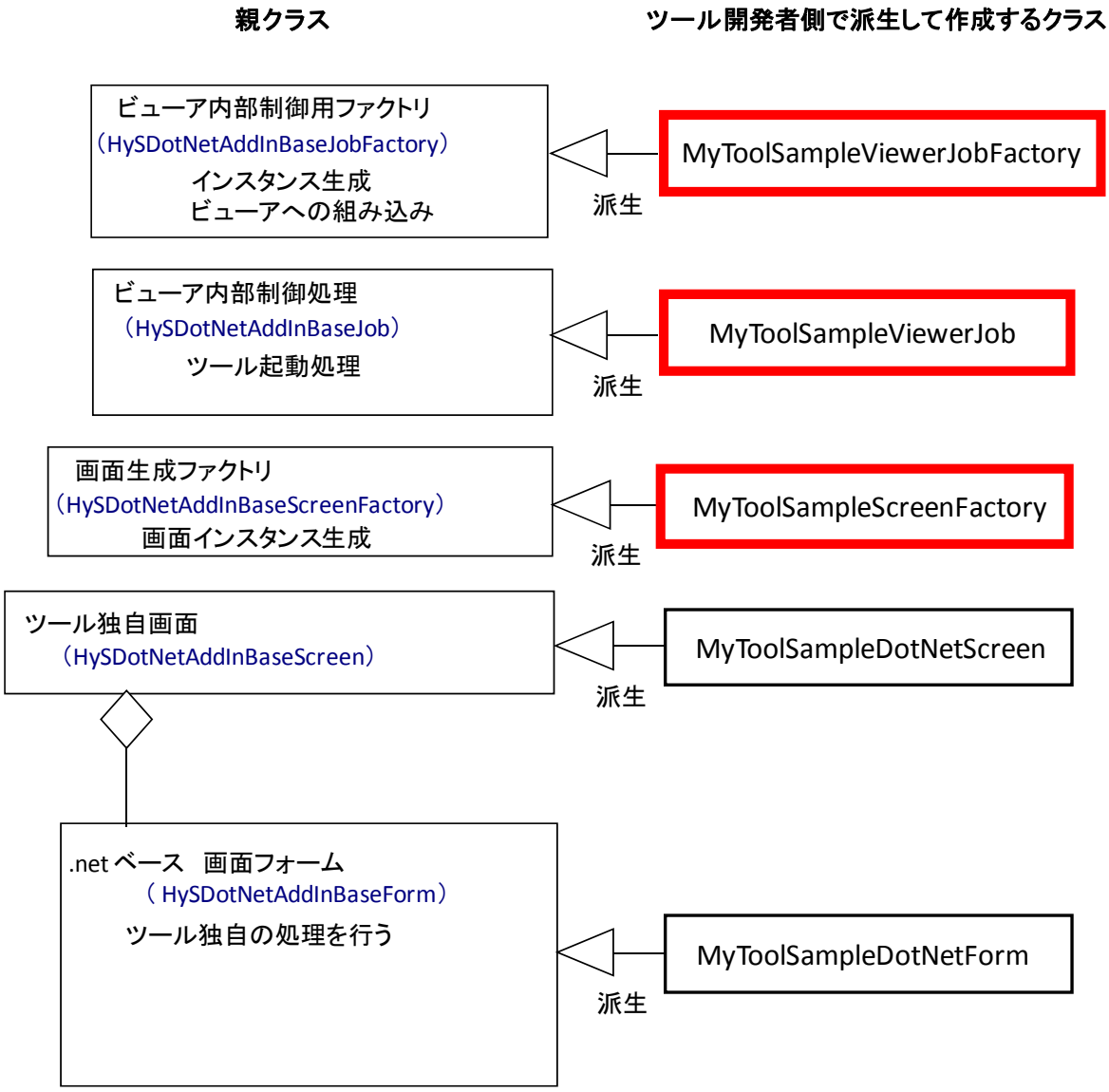

図 3.3　ツールの親クラスとツール開発者が作成すべきクラス

図 3.3 に示すツール開発者が作成するクラスの中で、CommonMPVer1.3 で追加される簡略化機能に対応する為に、ツール開発者にて対応（実装）が必要なクラスは、赤枠で囲まれているクラスとなります。以降、本クラスについて説明します。

4.　簡略化機能に対応する為にツール開発者が実装する処理

本章では、CommonMPVer1.3 で追加される下記の簡略化機能

●ツール本体を含むファイル一式をパッケージ化

●パッケージ単位で CommonMP 本体へインストール及び削除

●起動用のランチャーからツールを起動

に対応する為に実装する処理について説明します。

4.1　ビューア内部制御用ファクトリ（MyToolSampleViewerJobFactory）

CommonMP Ver1.2 以前では、本ファクトリに関する情報を CommonMP.dicon 内に定義しておく必要がありましたが、CommonMPVer1.3 以降では不要となります。

ツール開発者においては、ツール自身のファクトリクラスをフレームワーク側に取り込む処理を実装する必要があります。また、CommonMPVer1.3 以降ではツールの起動はランチャー画面から行うようになる為、起動に必要な情報を生成するメソッドを実装します。ランチャーに表示されるツール名称については、本サンプルのコーディング例に従い実装することで、OS 設定に応じて英語・日本語表示切り替えに対応することができます。

ビューア内部制御用ファクトリ
（HySDotNetAddInBaseJobFactory派生 MyToolSampleViewerJobFactory ）

ツール自身のファクトリクラスをフレームワーク側に取り込む処理

```
namespace CommonMP.HYSSOP._SYSTEM_
{
    public class HYSSOP_USRVIEWERJOB_FACTORY_CREATOR_IMPLE : HYSSOP_USRVIEWERJOB_FACTORY_CREATOR
    {
        public CommonMP.HYSSOP.Interface.HSViewer.HySViewerJobFactory CreateFactory()
        {
            // ToDo 自作の画面ファクトリ（ネームスペース付き）を new して下さい。
            return new jp.go.yyyy.MySysToolSample.MyToolSampleViewerJobFactory();
        }
    }
}
```

起動に必要な情報を生成するメソッド

```
override public HySUserToolInfoData CreateToolInf()
{
    HySString csToolName;
    //★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
    // ↓↓↓　多言語対応　↓↓↓
    //★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
    // ランチャー画面に表示されるツール名称をOS設定に応じて表示切替
    if (System.Threading.Thread.CurrentThread.CurrentUICulture.Name.StartsWith("ja") == true)
    {
        // OS設定が日本語の場合
        csToolName = new HySString("演算結果表示ツール");
    }
    else
    {
        // OS設定が日本語以外の場合
        csToolName = new HySString("Simulation Results Monitor Tool");
    }
    //★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
    // ↑↑↑　多言語対応　↑↑↑
    //★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
    //                          ユニークな値にして下さい。　ランチャーに表示される名称です。　ランチャーに表示されるアイコンです。
    //                                                    ↓                ↓                ↓
    return new HySUserToolInfoData(MyToolSampleDefine.BUSINESS_KIND, csToolName, new HySString("MySysToolSample.ico"));
}
```

図 4.1　ビューア内部制御用ファクトリで実装すべきメソッド

4.2　ビューア内部制御処理（MyToolSampleViewerJob）

CommonMPVer1.3 以降ではランチャー画面からボタンが押下されると、本メソッドがコールされます。ツール開発者はツール起動の処理を実装します。

本サンプルでは、ツール起動の処理としてツール独自画面を表示しています。

ビューア内部制御処理
（HySDotNetAddInBaseJob派生 MyToolSampleViewerJob）

ツール起動の処理を実装するメソッド

```
public override void StartTool()
{
    HySLog.LogOut(HySLog.SYSTEM_DEBUG, "MyToolSampleViewerJob", "StartTool", "画面表示開始"); // 必要に応じてログ出力

    HySID csID = null;
    csID = new HySID("ArbitraryCode");  // 画面には ユニークな値を一意に設定

    // 画面表示
    HySDotNetAddInBaseScreen csDotNetScreen
        = this.CmndShowScreen((HySObjectKind)MyToolSampleDefine.SAMPLE_SCREEN, csID) as HySDotNetAddInBaseScreen;
    if (csDotNetScreen != null)
    {   // 表示に成功ならば
        csDotNetScreen.ActivateForm();  // 表示した画面を最前面に持ってくる
    }

}
```

図 4.2　ビューア内部制御処理で実装すべきメソッド

4.3　画面生成ファクトリ（MyToolSampleScreenFactory）

本ファクトリについても 4.1　と同様、ツール自身のファクトリクラスをフレームワーク側に取り込む処理を実装する必要があります。

画面生成ファクトリ
（HySDotNetAddInBaseScreenFactory派生 MyToolSampleScreenFactory）

ツール自身のファクトリクラスをフレームワーク側に取り込む処理

```
namespace CommonMP.HYSSOP._SYSTEM_
{
    public class HYSSOP_USRSCREEN_FACTORY_CREATOR_IMPLE : HYSSOP_USRSCREEN_FACTORY_CREATOR
    {
        public CommonMP.HYSSOP.Interface.HSViewer.HySScreenFactory CreateFactory()
        {
            // ToDo 自作の画面ファクトリ（ネームスペース付き）を new して下さい。
            return new jp.go.yyyy.MySysToolSample.MyToolSampleScreenFactory();
        }
    }
}
```

図 4.3　画面生成ファクトリで実装すべきメソッド

5.　簡略化機能に対応する為にツール開発者が対応する内容

本章では、実装面以外で対応すべき内容について説明します。

5.1　設定する識別子の統一

CommonMPVer1.3 以降では、ツール開発者が設定・ユニークとすべき識別子を業務識別子のみとします。本サンプルでは業務識別子は、

¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥jp.go.xxx.MySysToolSample¥

下の MyToolSampleDefine.cs に定義されています。

Ver1.2(従来)	Ver1.3
ツール開発者が作成するツールにおいて 下記3種類の識別子を設定・ユニークとする ①業務識別子：開発者ごとに作成したツール（一種の業務処理）を識別する ②ファクトリ識別子 （ビューア内部制御用ファクトリ）：DLL内に含まれるファクトリを識別する ③ファクトリ識別子 （画面生成ファクトリ）： DLL内に含まれるファクトリを識別する	ツール開発者が作成するツールにおいて 下記**業務識別子のみ**を設定・ユニークとする **①業務識別子**：開発者ごとに作成したツール（一種の業務処理）を識別する

図 5.1　設定する識別子の統一

5.2　識別子の重複防止

前述の設定すべき識別子について、他の開発者ツールと重複しないように要素モデルと同様にドメイン名方式で定義します。本サンプルでは「jp.go.xxx.MySysToolSample」としています。

Ver1.2(従来)	Ver1.3
下記の識別子について、ツール開発者間でユニークとなるように開発者自身が任意に設定 ①業務識別子：任意に設定 ②ファクトリ識別子 （ビューア内部制御用ファクトリ）：任意に設定 ③ファクトリ識別子 （画面生成ファクトリ）：任意に設定	要素モデルと同様、**ドメイン名方式**により、ツール開発者間でユニークとなるよう規約を定める ①業務識別子： jp.go.XXX.ツール名

図 5.2　識別子の重複防止

5.3　ファイル等の重複防止

前述の識別子と同様に、ツールのファイル等についても、本ツールを利用する利用者のインストール環境において他のツールと重複しないように要素モデルと同様にドメイン名方式で定義します。

Ver1.2(従来)	Ver1.3
下記のファイルについて、ツール開発者間でユニークとなるように開発者自身が任意に設定 ①ツールのDLL名称：任意に設定	要素モデルと同様、**ドメイン名方式**により、ツール開発者間でユニークとなるよう規約を定める ①ツールのDLL名称： jp.go.XXX.ツール名.dll ※DLLから別なアプリケーションを起動する場合は、exeファイルの名称もユニークとする。 jp.go.XXX.アプリケーション名.exe ②アイコンファイル名称：←ツール開発者が任意で作成 jp.go.XXX.ツール名.ico ③解説書ファイル名称：←ツール開発者が任意で作成 jp.go.XXX.ツール名.pdf ④ツールプロジェクト一式を格納するフォルダ名 jp.go.XXX.ツール名

図 5.3　ファイル等の重複防止

5.4　ツールのプロジェクトフォルダ

以上の内容を踏まえて、ツールのプロジェクト一式を作成します。ツールの DLL 名称とツールのプロジェクフォルダ名称は同じにしてください。

・bin-Release フォルダ：ツールの本体(dll)及びツールから起動されるアプリケーション(exe)を格納します。

・ToolIcon フォルダ：必要に応じて、ランチャー画面に表示されるアイコンを格納します。

・ToolManual フォルダ：必要に応じて、ツールの解説書を格納します。

また、特定ツールごとに必要なファイル（例：Excel のテンプレートファイル等）については、このプロジェクフォルダ下に格納してください。後述するツールのパッケージ化において、本ツールのプロジェクトフォルダについても含めるオプションをつける必要があります。

図 5.4 にツールのインストール概要を示します。ツールのファイル等は図 5.4 に示すように実行する利用者の実行環境に格納されます。特定ツールごとに何かファイルを参照する必要がある場合は、本内容を理解しておくことが必要です。

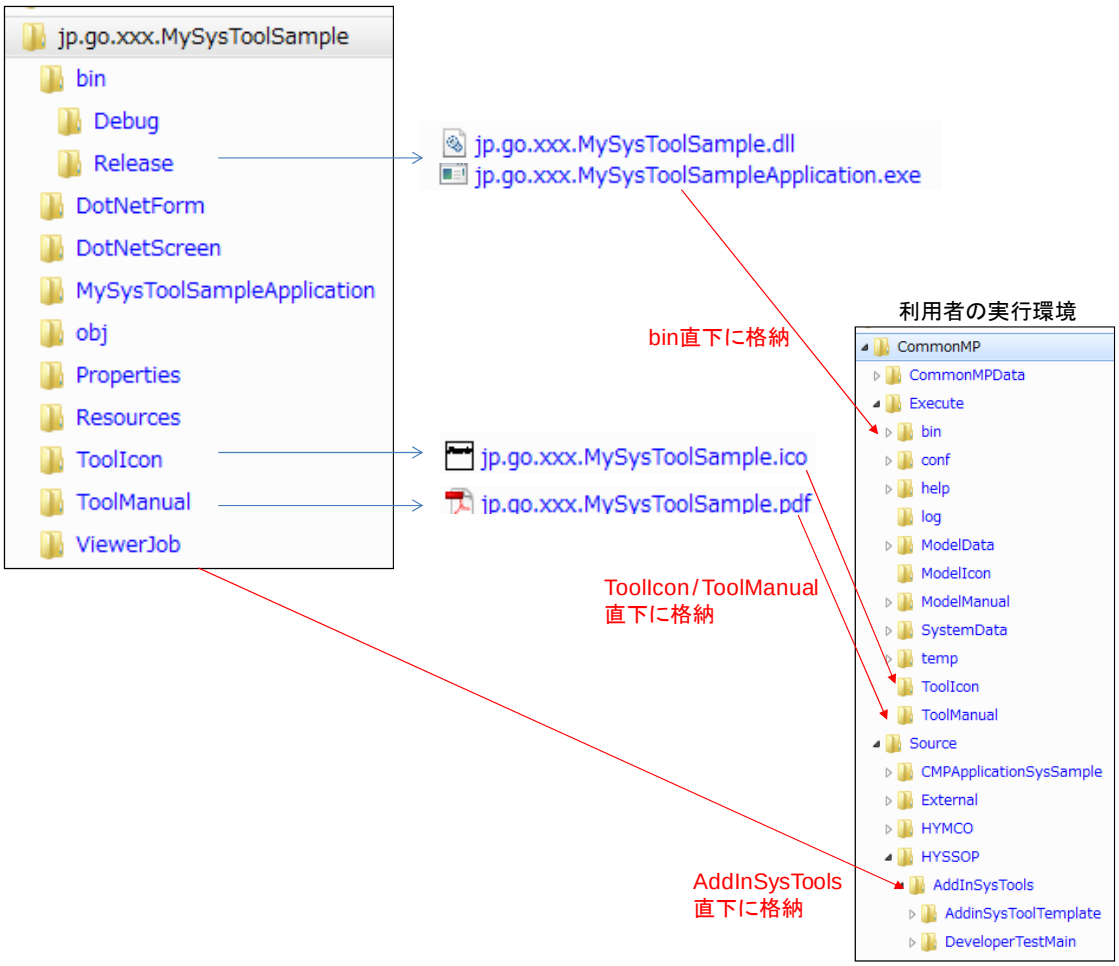

図 5.4　ツールのインストール概要

6. 配布準備

以上、CommonMPVer1.3 で追加される簡略化機能に対応する為の実装が完了し、特定ツール独自の機能の実装が完了すると配布に向けた準備を行います。

6.1 動作確認

¥CommonMP¥Source¥HYSSOP¥AddInSysTools¥下の「SysToolTestMain.sln」を、Visual Studio から開きます。ソリューションエクスプローラから SysToolTestMain の参照設定を右クリックして、表示されるメニューの「参照の追加」を選択します。

プロジェクトタブから開発したツールのプロジェクトを選択して、参照を追加しておきます。

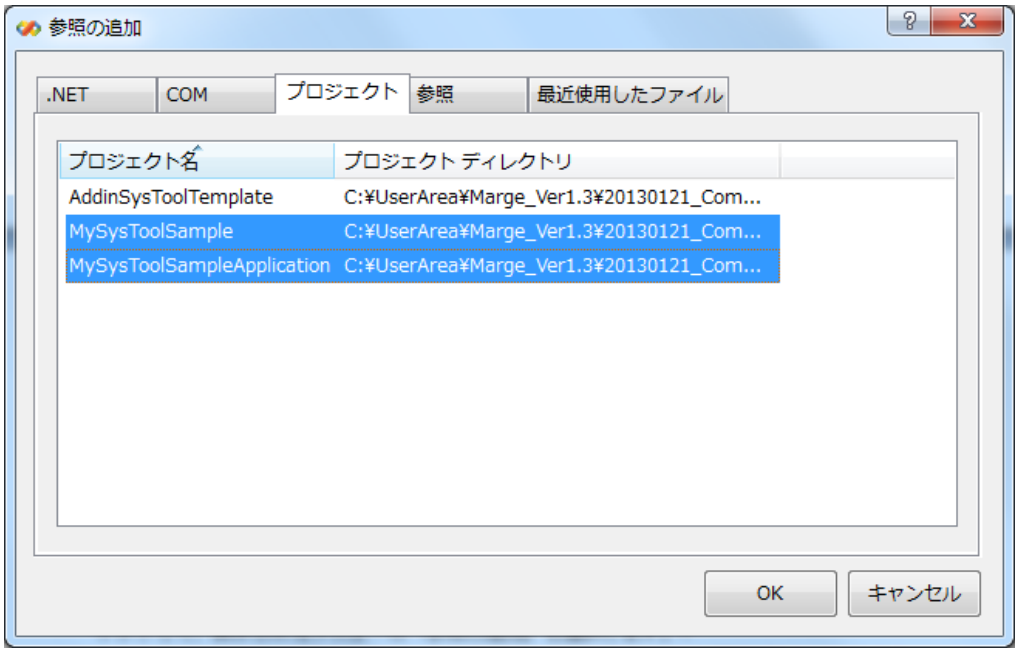

図 6.1　参照の追加

Visual Studio のメニュー「ビルド」－「ソリューションのリビルド」を実施後、

「デバッグ」－「デバッグ開始」を実行します。

CommonMP が起動しますので、メニューの「ヘルプ」－「ツール管理」－「ツール起動」を選択してランチャーにツールが表示されることを確認します。

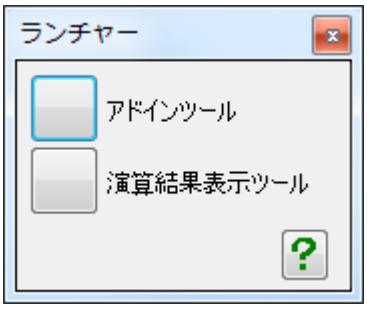

図 6.2　ランチャー起動

6.2　ツールのパッケージ化

CommonMPVer1.3 以降、ツールの配布にはパッケージ化が必要となります。

まず、ツールの実行モジュールを生成します。

6.2.1　ソリューション構成の変更

Visual Studio のツールバー上のソリューション構成を Debug から Release に変更します。

図 6.3　ソリューション構成の変更

Visual Studio のメニュー「ビルド」－「ソリューションのリビルド」を実施します。

エラーが無いことを確認します。

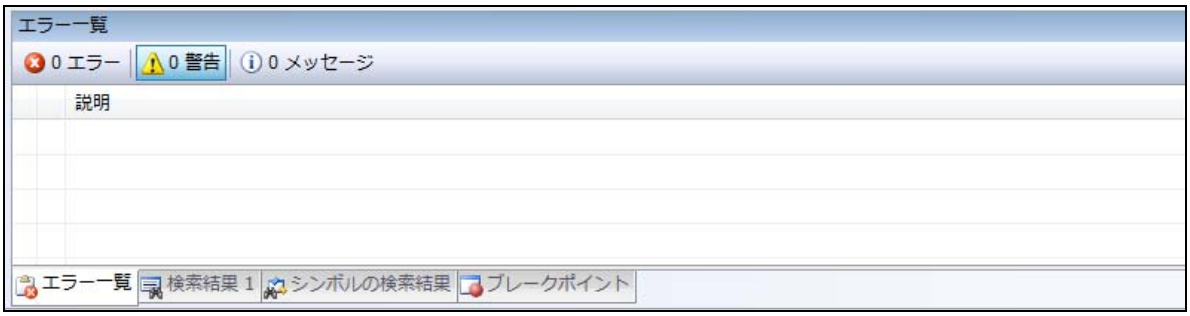

図 6.4　エラー一覧ウィンドウ

実施後は、Visual Studio のツールバー上のソリューション構成を Release から Debug に戻します。

6.2.2　パッケージ化

Visual Studio のメニュー「デバッグ」－「デバッグ開始」を実行します。

CommonMP が起動しますので、メニューの「ヘルプ」－「ツール管理」－「ツールのパッケージ化」を選択してツールのパッケージ化画面起動し、パッケージ化を行います。

なお、詳細な利用手順は、「CommonMP 操作手順書」を参照ください。

図 6.5　ツールのパッケージ化

ここで、特定ツールごとに必要なファイル（例：Excel のテンプレートファイル等）をツールのプロジェクフォルダ下に格納している場合は、必ず本画面の「開発ツールのプロジェクトフォルダを含める」にチェックしてください。