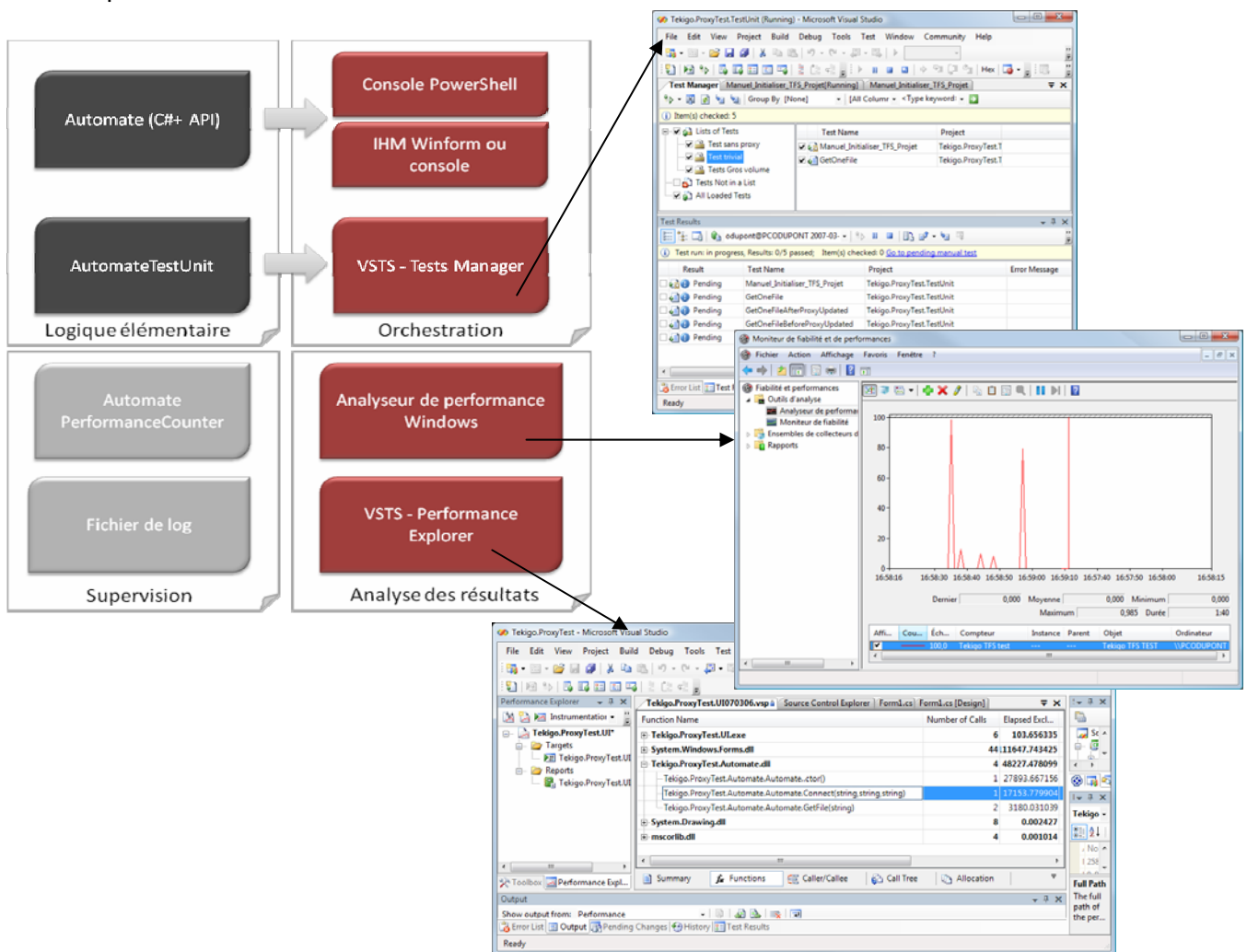


Voici un petit focus sur les possibilités de VSTS en matière de test. En effet, on peut se demander comment effectuer des mesures les plus proches possible de la réalité et cela sans y passer trop de temps. Afin de collecter les différentes métriques (délais, comptage...) nécessaires à l'analyse de l'utilisation du proxy, par exemple, il est possible de monter rapidement un automate grâce à Visual Studio 2005 et les API de Team Foundation Server. Selon la nature et la volumétrie des scénarios de tests à effectuer, ainsi que le détail de reporting souhaité, on peut choisir différentes solutions techniques aisément implémentables tels que :



Dans la démarche, on commence par écrire un automate (*helper*) implémentant les opérations élémentaires de manipulation du module *Source Control* de TFS (connexion, get, checkin, création de fichiers...) à l'aide des API de `Microsoft.TeamFoundation.VersionControl.client`. On peut ainsi orchestrer un certain nombre de scénarios via un script PowerShell ou via une IHM Winform. Afin d'être plus

flexible et induire une stratégie de plan de test plus ambitieuse et à faible coût, on peut s'appuyer sur le Test Manager de VSTS pour organiser des scénarios de tests unitaires complets. Une fois l'orchestration établie, il faut générer les métriques à l'exécution de l'automate. Pour cela, on s'appuie sur les API des capteurs de performances de Windows qui nous permettront de suivre en « temps réel » le comportement de l'automate, et ce de manière graphique. Ces mesures concernent l'usage effectif du TFSSC et se limitent à ce que l'on veut prouver. Pour avoir des informations complémentaires sur des opérations ne concernant pas directement les scénarios de tests, on pourra utiliser le Performance Explorer de VSTS pour isoler techniquement des délais d'exécution telle que la connexion, sans développement supplémentaire.

Exemple de code à implémenter :

```
[...]
public void Connect(String pTfsName, String pSourceControlPath, String pWorkspaceName)
{
    try
    {
        stopwatch.Start();
        //connexion à TFS
        _tfs = new TeamFoundationServer(pTfsName);
        // Connexion au SourceControl
        _versionControl = (VersionControlServer)_tfs.GetService(typeof(VersionControlServer));
        // Déclenchement capteur sur opération TFSSC
        _versionControl.OperationStarting += new OperationEventHandler(_versionControl_OperationStarting);
        _versionControl.OperationFinished += new OperationEventHandler(_versionControl_OperationFinished);
        // Configuration du Workspace
        _versionControl.DeleteWorkspace(pWorkspaceName, _versionControl.AuthenticatedUser);
        _workspace = _versionControl.CreateWorkspace(pWorkspaceName, _versionControl.AuthenticatedUser);
        _workspace.Map(pSourceControlPath, @"d:\temp\WorkspaceTekigo");
    }
    catch (Exception exception)
    {
        if (_workspace != null) _workspace.Delete();
        throw new Exception("Erreur de connexion", exception.InnerException);
    }
    finally
    {
        stopwatch.Stop();
        _log.Audit(String.Format("Temps écoulé : {0}", stopwatch.ElapsedMilliseconds));
        stopwatch.Reset();

        _tfs.Dispose();
    }
}

// operation VersionControl Start
void _versionControl_OperationStarting(object sender, OperationEventArgs e)
{
    stopwatch.Start();
    _log.Audit(DateTime.Now + ":" + e.Type.ToString() + " Started !");
}

// operation VersionControl Stop
void _versionControl_OperationFinished(object sender, OperationEventArgs e)
{
    stopwatch.Stop();
    // Enregistré le temps d'exécution dans le compteur de performance
    _performanceCounters[counterName].IncrementBy(stopwatch.ElapsedTicks);
    _performanceCounters[String.Concat(counterName, "Base").Increment();

    _log.Audit (DateTime.Now + ":" + e.Type.ToString() + " Finished !");
    _log.Audit(String.Format("Temps écoulé : {0}", stopwatch.ElapsedMilliseconds));
    stopwatch.Reset();
}

// Récupération d'un fichier du source control
public void GetFile(String pNameFile)
{
    GetRequest getRequest = new GetRequest(pNameFile, RecursionType.None, VersionSpec.Latest);
    GetStatus getStatus = _workspace.Get(getRequest, GetOptions.Overwrite);
    Failure[] failures = getStatus.GetFailures();
    _log.Info(String.Format("Nombre d'erreur : {0}", failures.Length.ToString()));
    foreach (Failure failure in failures)
    {
        _log.Error(String.Format("Status : {0}|{1}|{2}", failure.Severity, failure.LocalItem,
failure.Message));
    }
}
[...]
```