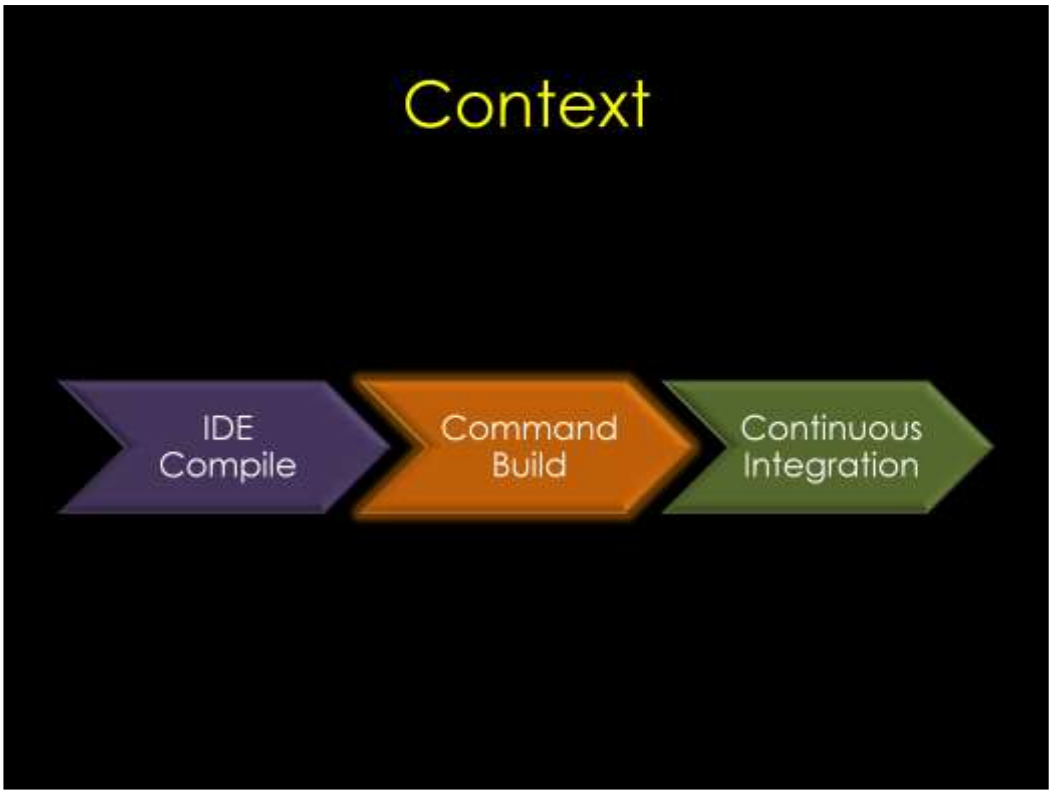

Nant
Bevan Arps
www.nichesoftware.co.nz


Context
IDE
Compile
Command
Build
Continuous
Integration

Simple target to compile our assemblies

```
 <target name="compile.assemblies"
       description="Build output from source”>

   <exec
program="C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\msbuild.exe">
     <arg file="NicheDashboard.sln"/>
     <arg value="/t:rebuild"/>
     <arg value="/verbosity:quiet"/>
   </exec>

 </target>
```

Instead of hard coding the location of MSBuild, set up a dedicated target to check

```
<target name="require.msbuild"
      description="Ensure MSBuild is available.">

  <!-- Define the directory where we expect to find msbuild -->
  <property name="msbuild.dir"
        value="C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319"/>

  <property name="msbuild.exe"
        value="${ path::combine( msbuild.dir, 'msbuild.exe' ) }"/>

  <fail message="Didn't find ${msbuild.exe}"
      unless="${ file::exists( msbuild.exe ) }"/>

</target>
```

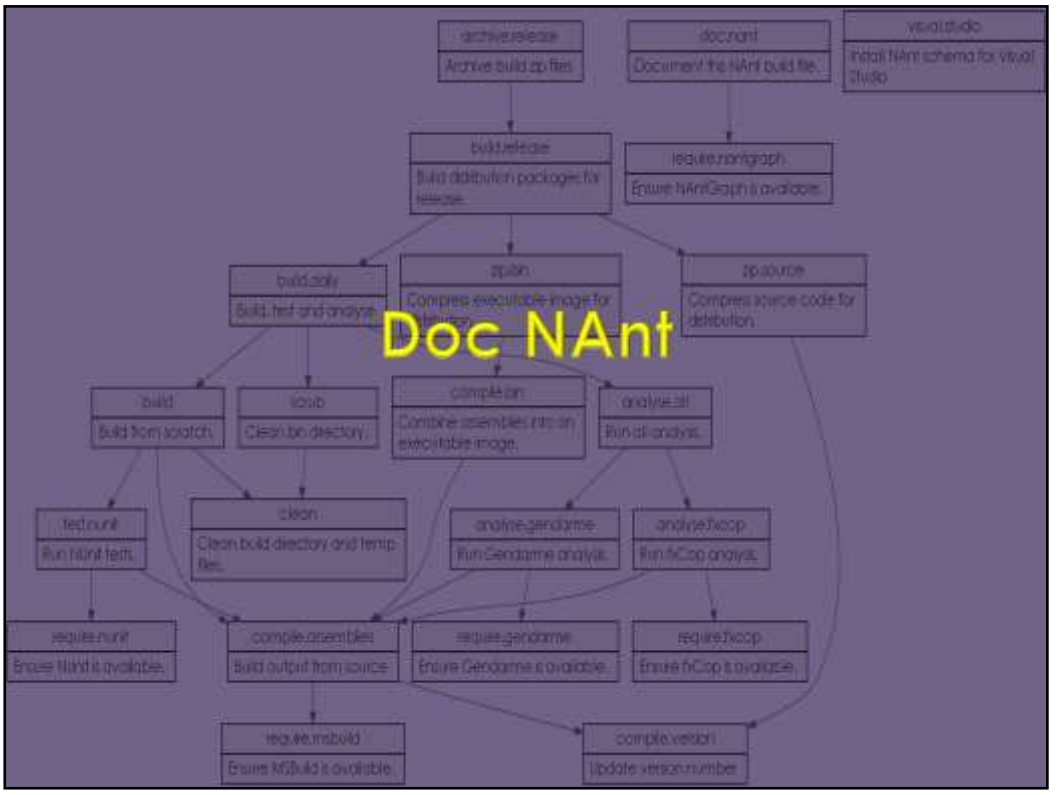
Update compile.assemblies to depend on the new target

Add a target to clean up

```
<target name="clean" description="Clean build directory and temp files.">

  <delete>
    <fileset basedir="${build.dir}">
      <include name="**/*"/>
      <exclude name="**/*.vshost.exe"/>
    </fileset>
  </delete>

  Special handling for the visual studio host
  <delete failonerror="false" >
    <fileset basedir="${build.dir}">
      <include name="**/*.vshost.exe"/>
    </fileset>
  </delete>

  <delete>
    <fileset basedir="${tmp.dir}">
      <include name="**/*"/>
    </fileset>
  </delete>

</target>
```

Combine all the stops so far into a simple build target

```
<target name="build"
      depends="clean compile.assemblies test.nunit"
      description="Build from scratch."/>
```



Ensure NAntGraph is available

```
 <target name="require.nantgraph"
       description="Ensure NAntGraph is available.">

   <!-- Define the directory where we expect to find msbuild -->
   <property name="nantgraph.dir"
          value="${lib.dir}\nantgraph"/>

   <property name="nantgraph.exe"
          value="${ path::combine( nantgraph.dir, 'NAntGraph2.exe' ) }"/>

   <fail message="Didn't find ${nantgraph.exe}"
       unless="${ file::exists( nantgraph.exe ) }"/>

 </target>
```

Generate some documentation of this build file

```
 <target name="compile.nantgraph"
       depends="require.nantgraph"
       description="Document this NAnt build file.">

   <mkdir dir="${report.dir}"/>

   <exec program="${nantgraph.exe}">

     <arg value="${base.dir}\dashboard.build"/>
     <arg value="--font=Century Gothic"/>
     <arg value="--fontsize=16"/>
     <arg value="--descriptions"/>
     <arg value="--out=${report.dir}\dashboard.png"/>

   </exec>

 </target>
```

Lets set up integrated versioning across all our assemblies.

Define the version number.

```
 <target name="compile.version” description="Update version number">

   <version startdate="1 May 2010"
         path="${src.dir}\version.txt"
         prefix="build"/>

 </target>
```

Add target to build a versioninto file

```
 <target name="compile.versioninfo“
       description=“Build VersionInfo file“
       depends=“compile.version”>

   <asminfo language="CSharp" output="src\VersionInfo.cs">
     <imports>
       <import namespace="System" />
       <import namespace="System.Reflection" />
     </imports>
     <attributes>
       <attribute type="AssemblyVersionAttribute" value="${build.version}" />
       <attribute type="AssemblyFileVersionAttribute" value="${build.version}" />
     </attributes>
</target>
```

Add link to compile.assemblies

Define a target to check that Nunit is available

```
<target name="require.nunit“ description="Ensure NUnit is available.">

  <!-- Define the directory where we expect to find msbuild -->
  <property name="nunit.dir“ value="${lib.dir}/nunit"/>

  <property name="nunit.exe“ value="${ path::combine( nunit.dir, 'nunit-console.exe' ) }"/>

  <fail message="Didn't find ${nunit.exe}“ unless="${ file::exists( nunit.exe ) }"/>

</target>
```

*Run all our Nunit tests – we look for all *TESTS.dll to automatically pick up new tests*

```
<target name="test.nunit” depends="require.nunit compile.assemblies“description="Run NUnit tests.">

  <property name="assemblies" value=""/>
  <foreach item="File" property="file">
    <in>
      <items basedir="${debug.dir}">
        <include name="**/*.Tests.dll"/>
      </items>
    </in>
    <do>
      <property name="assemblies" value="${assemblies} ${file}"/>
    </do>
  </foreach>

  <exec program="${nunit.exe}" failonerror="false">
    <arg line="${assemblies}"/>
    <arg value="/nodots"/>
  </exec>

</target>
```

Set up a target to check that fxcop is available

```
<target name="require.fxcop“ description="Ensure fxCop is available.">

  <!-- Define the directory where we expect to find msbuild -->
  <property name="fxcop.dir“ value="C:\Program Files\Microsoft FxCop 1.36"/>

  <property name="fxcop.exe“ value="${ path::combine( fxcop.dir, 'fxcopcmd.exe' ) }"/>

  <fail message="Didn't find ${fxcop.exe}“ unless="${ file::exists( fxcop.exe ) }"/>

  <property name="fxcop.res“ value="${res.dir}\fxcop"/>

</target>
```

Run the fxcop analysis

```
<target name=“compile.fxcop"
      depends="require.fxcop compile.assemblies"
      description="Run fxCop analysis.">

  <property name="fxcop.project” value="NicheDashboard.FxCop"/>

  <mkdir dir="${report.dir}"/>

  <exec program="${fxcop.exe}">
    <arg value="/project:${fxcop.project}"/>
    <arg value="/out:${report.dir}/fxcop.report.html"/>
    <arg value="/outxsl:${fxcop.res}/fxcop.xsl"/>
    <arg value="/applyoutxsl"/>
  </exec>

  <copy todir="${report.dir}“ file="${fxcop.res}/fxcop.css"/>

</target>
```

Image: http://www.flickr.com/photos/thomashawk/3147786573/

Gendarme is an open source alternative to fxcop. First, a target to ensure it is available

```
 <target name="require.gendarme" description="Ensure Gendarme is available.">

   <!-- Define the directory where we expect to find msbuild -->
   <property name="gendarme.dir" value="${lib.dir}\gendarme"/>

   <property name="gendarme.exe" value="${ path::combine( gendarme.dir, 'gendarme.exe' ) }"/>

   <fail message="Didn't find ${gendarme.exe}" unless="${ file::exists( gendarme.exe ) }"/>

   <property name="gendarme.res" value="${res.dir}\gendarme"/>

 </target>
```

Run the analysis

```
 <target name="compile.gendarme" depends="require.gendarme compile.assemblies" description="Run Gendarme analysis.">

   <mkdir dir="${report.dir}"/>

   <property name="assemblies" value=""/>
   <foreach item="File" property="file">
     <in>
       <items basedir="${debug.dir}\NicheDashboard">
         <include name="Niche*.dll"/>
         <include name="Niche*.exe"/>
         <exclude name="*vshost*"/>
       </items>
     </in>
     <do>
       <property name="assemblies" value="${assemblies} ${file}"/>
     </do>
   </foreach>

   <exec program="${gendarme.exe}" failonerror="false" >

     <arg value="--html"/>
     <arg value="${report.dir}\gendarme.html"/>
     <arg value="--ignore"/>
     <arg value="${src.dir}\gendarme.ignore"/>
     <arg line="${assemblies}"/>

   </exec>

   <copy todir="${report.dir}"
         file="${fxcop.res}/fxcop.css"/>

 </target>
```

Image: http://www.flickr.com/photos/billkatygemma/4258075702/

Perform a daily build

```
<target name="build.daily"
      description="Build, test and analyse"
      depends="scrub build test.nunit compile.fxcop compile.gendarme"/>
```


```
<target name="compile.image"
        depends="compile.assemblies"
        description="Combine assemblies into an executable image.">

   <copy todir="${bin.dir}">
    <fileset basedir="${debug.dir}\NicheDashboard">
     <include name="**"/>
    </fileset>
   </copy>

 </target>
```

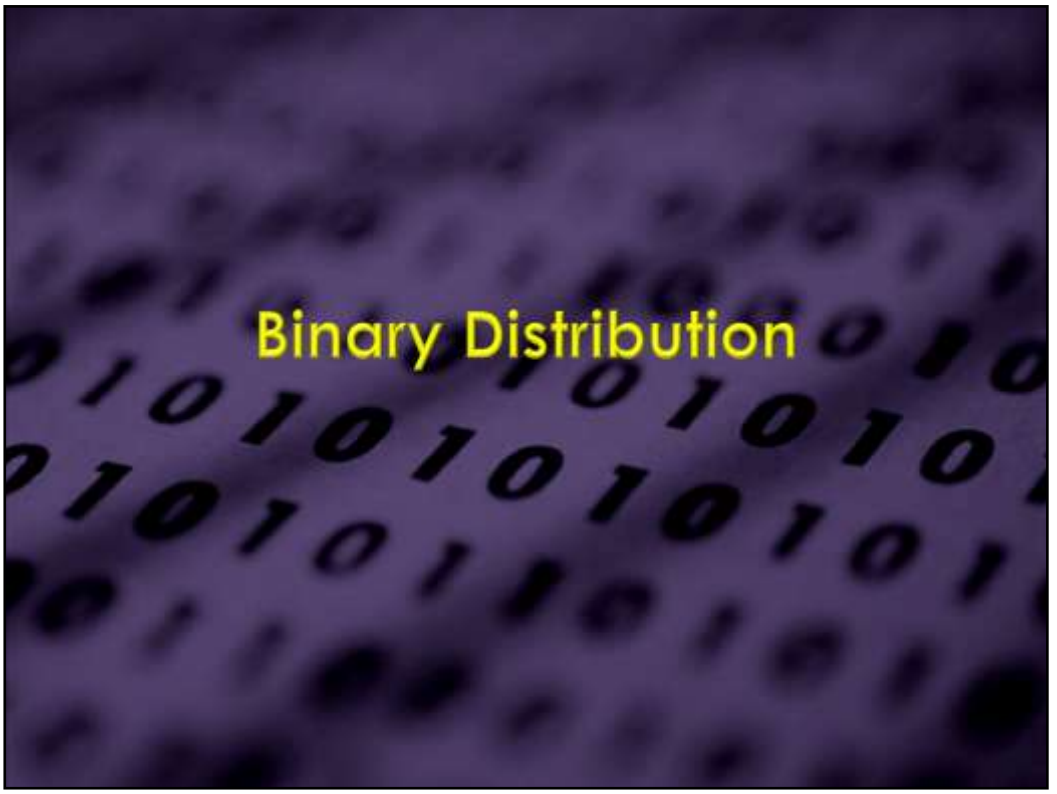
Image: http://www.flickr.com/photos/29609591@N08/4469019265/

A target to clean some more stuff

```
<target name="scrub"
        depends="clean"
        description="Clean bin directory.">

  <delete>
    <fileset basedir="${bin.dir}">
      <include name="**/*"/>
    </fileset>
  </delete>

</target>
```

Compress source for distribution

```
 <target name="zip.source"
       depends="compile.version"
       description="Compress source code for distribution.">

   <zip zipfile="${release.dir}\NicheDashboard-${build.version}-src.zip"
    ziplevel="9">
     <fileset basedir="${base.dir}">
       <include name="**"/>
       <exclude name="release/**"/>
       <exclude name="build/**"/>
       <exclude name="**/*.user"/>
       <exclude name="**.suo"/>
       <exclude name="**.cache"/>
       <exclude name="_ReSharper.NicheDashboard/**"/>
       <exclude name="**/obj/**"/>
       <exclude name="**/bin/**"/>
     </fileset>
   </zip>

 </target>
```



Compress the image for distribution

```
<target name="zip.bin"
      depends="compile.image"
      description="Compress executable image for distribution.">

  <zip zipfile="${release.dir}\NicheDashboard-${build.version}-bin.zip"
   ziplevel="9">
    <fileset basedir="${bin.dir}">
      <include name="**"/>
    </fileset>
  </zip>

</target>
```


```
<target name="build.distribution"
        depends="build.daily zip.source zip.bin"
        description="Build distribution packages for upload."/>
```


```
<target name="archive.release"
        depends="build.release"
        description="Archive build zip files">

   <property name="release.archive.dir"
           value="${ path::combine( archive.dir, build.version) }"/>

   <copy todir="${release.archive.dir}">
     <fileset basedir="${release.dir}">
       <include name="*{build.version}*.zip"/>
     </fileset>
   </copy>

 </target>
```


```
<!-- ============================================================================
     Visual Studio
    ============================================================================
     Install a schema for editing NAnt files into whatever versions of Visual
     Studio are currently installed.
    ============================================================================
 -->

 <target name="visual.studio"
       description="Install NAnt schema for Visual Studio">

  <property name="visual.studio.2008.dir"
         value="C:\Program Files\Microsoft Visual Studio 9.0"/>

  <property name="visual.studio.2010.dir"
         value="C:\Program Files\Microsoft Visual Studio 10.0"/>

  <nantschema output="nant.xsd"
          target-ns="http://nant.sf.net/release/0.85/nant.xsd"/>

  <copy file="nant.xsd"
      todir="${visual.studio.2008.dir}\Xml\Schemas\"
      if="${ directory::exists( visual.studio.2008.dir ) }"/>

  <copy file="nant.xsd"
      todir="${visual.studio.2010.dir}\Xml\Schemas\"
      if="${ directory::exists( visual.studio.2010.dir ) }"/>

  <delete file="nant.xsd"/>

</target>
```

Any questions kept until the end?


Bevan Arps
bevan@nichesoftware.co.nz
http://www.nichesoftware.co.nz