APT Builds Requirements

- Support all current development platforms.
- Ability to individually build sub-systems.
- Ability to do build maintenance tasks globally on all sub-systems:
    - **Build:** Generates class files and gathers resources under the sub-system sub-directories. If **Build** fails on a sub-system, it must be able to continue building any remaining sub-systems.
    - **Deliver:** Delivers class files and resources generated with **Build** to the appropriate delivered places such as the Libraries or Plug-Ins sub-directories. The files for each sub-system are built in their dependency order.
    - **Clean:** Deletes class files and resources generated by **Build.**
    - **CleanDelivery:** Deletes files delivered by **Deliver.**
    - **CleanAll:** Performs **Clean** and **CleanDelivery**
    - **Rebuild:** Performs **Clean** followed by **Build**.
    - **Redeliver:** Performs **CleanAll** followed by **Deliver**.
    - **JavaDoc:** Generates JavaDoc for all sub-systems.
- "Easy" to maintain.
- Allow overnight builds to send email if and only if it fails.

APT Build Design

- **<u>High Level Design</u>**
  - Maintain a build mechanism for each sub-system.
  - Build mechanisms support a minimum set of *build*:

    build, deliver, clean, cleandelivery, cleanall, rebuild, redeliver
  - Maintain a top-level build mechanism which allows *build tasks* to be executed on the sub-systems in the appropriate dependency order. Also includes a **javadoc** task to generate Java Doc for the whole system in APT/API.
  - Replace Unix *build* and *makejar* scripts with *ANT* to implement the build mechanism.
- ***<u>ANT</u>*:** An open source java-based system for maintaining resources that depend upon one another. Basically, implements the functionality of *make* in java and uses XML files for defining the dependencies. It can also work as a *very, <u>very</u>*, ***<u>very</u>*** simple platform independent scripting language.

```
<project name="VisitPlanner" default="build" basedir=".">

     <property name="src"         value="${basedir}"/>
     <property name="classdir"    value="${basedir}/classes"/>
     <property name="jarNameRoot" value="${ant.project.name}"/>
     <property name="classesJar"  value="${jarNameRoot}.jar"/>
     <property name="sourceJar"   value="${jarNameRoot}-Source.jar"/>

     <property name="apt"         value="${basedir}/.."/>
     <property name="libraries"   value="${apt}/Libraries"/>
     <property name="plugins"     value="${apt}/Plug-Ins"/>

     <property name="voltFiles"  value="${basedir}/voltFiles"/>
     <property name="voltJars"   value="${voltFiles}/jarFiles"/>
<!--
   ================================================================================
   ================================================================================
```

```
   Set deliverClassesJarTo value to the delivery place for jar files.  It
   should be either ${libraries} or ${plugins}.  Note, that the source jar
   file will be delivered to ${deliverClassesJarTo}/Source.
   ============================================================================
-->
     <property name="deliverClassesJarTo" value="${plugins}"/>

     <property name="deliverSourceJarTo"
        value="${deliverClassesJarTo}/Source"/>
<!--
   ============================================================================
   ============================================================================
   The value of sources is a comma delimited list of the source directories
   such is "edu/**,gov/**".
   ============================================================================
-->
     <property name="sources" value="edu/**,gov/**"/>
<!-- ============================================================================
 -->
<!--
   ============================================================================
   ============================================================================
   Make buildClasspath semi-colon delimited.  It should include ${classdir} and
   only those jars necessary for building this subsystem
   ============================================================================
-->
     <property name="buildClasspath"
        value="
         ${classdir};
           ${libraries}/APT.jar;
           ${libraries}/Interfaces.jar;
           ${libraries}/jdom.jar;
           ${libraries}/jh.jar;
           ${libraries}/jsky-overrides.jar;
           ${libraries}/jsky.jar;
           ${libraries}/SEA.jar;
           ${libraries}/Utilities.jar;
           ${voltJars}/bsf.jar;
           ${voltJars}/class_server.jar;
           ${voltJars}/jmf.jar;
           ${voltJars}/jpython.jar;
           ${voltJars}/js.jar;
           ${voltJars}/jsdt.jar;
           ${voltJars}/JSolver206.zip;
           ${voltJars}/sun_ext.jar;
           ${voltJars}/volt.jar
     "/>
<!-- ============================================================================
 -->
<!--
   ============================================================================
   ============================================================================
   Add any tasks to the init target which should be done before anything else.
   ============================================================================
-->
     <target name="init">
        <tstamp/>
```

```
        <mkdir dir="${classdir}"/>
      </target>
<!-- ================================================================================
 -->
<!--
   ================================================================================
   ================================================================================
   Customize how the subsystem is built in the compile target.
   ================================================================================
-->
     <target name="compile" depends="init">
        <javac srcdir="${src}" optimize="on" destdir="${classdir}"
              includes="${sources}">
           <classpath path="${buildClasspath}"/>
        </javac>
     </target>
<!-- ================================================================================
 -->
<!--
   ================================================================================
   ================================================================================
   Customize how the subsystem is built in the compiledebug target.
   ================================================================================
-->
     <target name="compiledebug" depends="init">
        <javac srcdir="${src}" debug="on" destdir="${classdir}"
              includes="${sources}">
           <classpath path="${buildClasspath}"/>
        </javac>
     </target>
<!-- ================================================================================
 -->
<!--
   ================================================================================
   ================================================================================
   This copies resources (all files except java and CVS) in the source trees
   to the classes tree.  You can add more copy tasks to copy other files to the
   classes tree so they'll be jarred up at delivery time.
   ================================================================================
-->
     <target name="copyresources" depends="init">
        <copy todir="${classdir}/edu">
           <fileset dir="${basedir}/edu">
            <exclude name="**/*.java"/>
            <exclude name="**/CVS"/>
         </fileset>
      </copy>
        <copy todir="${classdir}/gov">
           <fileset dir="${basedir}/gov">
            <exclude name="**/*.java"/>
            <exclude name="**/CVS"/>
         </fileset>
      </copy>
        <copy todir="${classdir}">
           <fileset dir="${voltFiles}">
            <exclude name="**/*.java"/>
            <exclude name="**/CVS"/>
```

```
            <exclude name="**/*.jar"/>
            <exclude name="jarFiles/**"/>
         </fileset>
        </copy>
      <copy todir="${classdir}" file="${basedir}/MANIFEST.MF"/>
     </target>
<!-- ============================================================================
 -->
<!--
   ============================================================================
   ============================================================================
   The building, delivering and cleaning targets shouldn't be modified as the
   top level build.xml file for APT depends upon them.
   ============================================================================
-->
     <target name="build" depends="compile,copyresources"/>

     <target name="builddebug" depends="compiledebug, copyresources"/>

     <target name="rebuild" depends="clean,build"/>

     <target name="rebuilddebug" depends="clean,builddebug"/>

     <target name="deliver" depends="build">
        <jar jarfile="${deliverClassesJarTo}/${classesJar}"
           basedir="${classdir}"/>
        <jar jarfile="${deliverSourceJarTo}/${sourceJar}"
           basedir="${basedir}" includes="${sources}"/>
     </target>

     <target name="redeliver" depends="cleanall,deliver"/>

     <target name="clean" depends="">
        <mkdir dir="${classdir}"/>
        <delete includeEmptyDirs="true">
           <fileset dir="${classdir}" excludes="CVS"/>
        </delete>
      <delete file="${classesJar}"/>
      <delete file="${sourceJar}"/>
     </target>

     <target name="cleandelivery">
        <delete file="${deliverClassesJarTo}/${classesJar}"/>
        <delete file="${deliverSourceJarTo}/${sourceJar}"/>
     </target>

     <target name="cleanall" depends="clean,cleandelivery"/>
<!-- ============================================================================
 -->
     <property name="runtimeClasspath"
        value="
         ${classdir};
           ${libraries}/advisor.jar;
           ${libraries}/APT.jar;
           ${libraries}/AptDocumentModel.jar;
           ${libraries}/APTIDServer.jar;
           ${libraries}/crimson.jar;
```

```
        ${libraries}/fits.jar;
        ${libraries}/hcompress.jar;
        ${libraries}/HstDocumentModel.jar;
        ${libraries}/images.jar;
        ${libraries}/Interfaces.jar;
        ${libraries}/iText.jar;
        ${libraries}/jai_codec.jar;
        ${libraries}/jai_core.jar;
      ${libraries}/jakarta-regexp-1.2.jar;
        ${libraries}/jaxp.jar;
        ${libraries}/jcchart.jar;
        ${libraries}/jcommon.jar;
        ${libraries}/jdom.jar;
        ${libraries}/jess.jar;
        ${libraries}/jfreechart.jar;
        ${libraries}/jh.jar;
        ${libraries}/jsky-overrides.jar;
        ${libraries}/jsky.jar;
        ${libraries}/jspike.jar;
        ${libraries}/koala.jar;
        ${libraries}/mlibwrapper_jai.jar;
        ${libraries}/MRJToolkitStubs.zip;
        ${libraries}/OCM.jar;
        ${libraries}/SEA.jar;
        ${libraries}/ToolInterface.jar;
        ${libraries}/Utilities.jar;
        ${libraries}/xalan.jar;
        ${libraries}/xml.jar;
        ${voltJars}/bsfengines.jar;
        ${voltJars}/class_server.jar;
        ${voltJars}/fits.jar;
        ${voltJars}/jacl.jar;
        ${voltJars}/jhall.jar;
        ${voltJars}/jmf.jar;
        ${voltJars}/jpython.jar;
        ${voltJars}/js.jar;
        ${voltJars}/jsdt.jar;
        ${voltJars}/jsdt-client-socket.jar;
        ${voltJars}/jsky.jar;
        ${voltJars}/JSolver206.zip;
        ${voltJars}/mask.jar;
        ${voltJars}/netscape.jar;
        ${voltJars}/bsf.jar;
        ${voltJars}/sun_ext.jar;
        ${voltJars}/tcljava.jar;
        ${voltJars}/volt.jar;
        ${voltJars}/xml.jar;
  "/>

<target name="runsample" depends="deliver">
   <java
    classname="edu.stsci.visitplanner.VpSampleTool"
    fork="yes"
    classpath="${runtimeClasspath}">

     <arg value="v"/>
```

```
      </java>
   </target>

   <target name="runapt" depends="deliver">
      <java
       fork="yes"
       classpath="${runtimeClasspath}"
       classname="edu.stsci.hst.apt.controller.HstAptController"
       dir="${apt}">
      </java>
   </target>

   <target name="runsampledebug" depends="deliver">
      <java
       classname="edu.stsci.visitplanner.VpSampleTool"
       fork="yes"
       classpath="${runtimeClasspath}">

        <arg value="t"/>

      </java>
   </target>
<!-- ======================================================================
 -->
</project>
```

- **<u>Complications</u>**
  - **<u>clean</u>:** It is not easy to sort out files that need to be cleaned *versus* those that don't. The **clean** tasks have been designed to delete all files under the *classes* directories except for the CVS directories. So, **DO NOT COMMIT FILES UNDER THE *CLASSES* DIRECTORIES** unless you are prepared to re-implement the **clean** targets.
  - **<u>redeliver</u>:** For some reason I don't undertand, when the top level **redeliver** is implemented to execute **redeliver** on each sub-system, it fails on the nth sub-system when attempting to do **cleanall**. A work around is for the top level **redeliver** to do **cleanall** on all the sub-systems first, then do **deliver**.
  - **<u>Complex Top Level Build File</u>:** To set up dependencies among the sub-systems, the top level build file must have one target for each sub-system/target combination (7 x 17) plus the top level *build*

targets. Adding sub-systems or targets to the top level build file is tedious.

- **build:** When a target fails, *ANT* will halt.
- **idl:** There is no *idl* task in *ANT* which is needed by the Orbit Planner.

- **Complex Top Level Build File**

  If it's hard to do something, don't. So, instead of maintaining all the targets and dependencies in the top build file, this stuff is dynamically generated by a java program called *BuildFileGenerator.*

  The actual top level build file maintains *proxy* targets to the generated build file. Each of the proxy targets depends on a custom *ANT* task that invokes *BuildFileGenerator* so the actual build file is re-generated as needed.

  A list of the supported sub-systems, their dependencies and the *build targets* is maintained in an XML file which serves as the input to *BuildFileGenerator.*

  The 1600 lines required to maintain the build targets and dependencies reduces to about 200.

- **build:** Normally, *ANT* can be invoked as a task within *ANT* to run targets on the sub-systems. But, if you do this, *ANT* will halt if a sub-system fails to build.

  A **buildNoFail** target is used to invoke *ANT* as an external program for which failures *can* be trapped.

  However, this adds another complication. To invoke *ANT* externally, the path to the *ANT* program file is needed and this is operating system dependent. For systems with Java property "os.name" = "SunOS", the program file is named "ant". For "Windows…" it is "ant.bat". *BuildFileGenerator* assumes for all other systems that it is "ant".

- **idl:** The Orbit Planner build file is setup to invoke the java idl program as an external program. A *proxy* file is used to track the build date of the generated java files so they will be built only when necessary.

APT Build Code

Build File Data Fragments (APT/AntBuildData.xml):

```
    <Subsystem name="APT">
      <TargetsWithDependencies>
        <Target name="deliver">
          <Dependency>JSky</Dependency>
          <Dependency>OCM</Dependency>
        </Target>
      </TargetsWithDependencies>
    </Subsystem>

    <Subsystem name="APTIDServer"/>

  <Targets>
    <Target>build</Target>
    <Target>clean</Target>
    <Target>cleanall</Target>
    <Target>cleandelivery</Target>
    <Target>deliver</Target>
    <Target>rebuild</Target>
  </Targets>

  <CompoundTargets>
    <CompoundTarget name="redeliver">
      <Subtarget>cleanall</Subtarget>
      <Subtarget>deliver</Subtarget>
    </CompoundTarget>
  </CompoundTargets>
```

Top Level Build File Fragments (APT/build.xml):

```
  <target name="build" depends="generatebuildfile">
    <ant target="build"
         antfile="${generatedBuildFile}"
         dir="${basedir}"
         inheritall="true"
    />
  </target>

  <target name="buildFileGenerator">
    <ant target="deliver" dir="${basedir}/CM" inheritall="true" />
  </target>

  <target name="declareGenerateBuildFileTask" depends="buildFileGenerator">
    <taskdef name="generateBuildFile"
             classname="edu.stsci.apt.BuildFileGenerator"
             classpath="${basedir}/Libraries/Internal/CM.jar
                      :${basedir}/Libraries/jakarta-regexp-1.2.jar
                      :${basedir}/Libraries/crimson.jar
```

```
                    :${basedir}/Libraries/jdom.jar
                    :${env.ANT_HOME}/lib/ant.jar"
  />
</target>

<target name="generatebuildfile" depends="declareGenerateBuildFileTask">
  <generateBuildFile outBuildFile="${generatedBuildFile}"
                     buildDataFile="${basedir}/AntBuildData.xml"
  />
</target>
```

BuildFileGenerator Fragments (APT/CM/edu/stsci/apt/BuildFileGenerator.java):

```
public class BuildFileGenerator extends org.apache.tools.ant.Task

public final void execute() throws BuildException
{
      try
      {
            createBuildFile(fOutBuildFile, fBuildDataFile);

      } // try
      catch (Throwable iThrowable)
      {
            throw new BuildException(iThrowable);

      } // catch
} // execute()

public final void setBuildDataFile(File iFile)
{
      fBuildDataFile = iFile;

} // setBuildDataFile(File)


public final void setOutBuildFile(File iFile)
{
      fOutBuildFile = iFile;

} // setOutBuildFile(File)
```

Subystem Build Files (APT/<subsystem name>/build.xml):

See Visit Planner build file above.

Generated Build File Fragments (APT/.generatedBuildFile.xml):

```
<target name="buildAPT"
        depends="">
```

```
      <ant dir="${basedir}/APT"
           target="build"
      />
   </target>

   <target name="build"

depends="buildAPT,buildAPTIDServer,buildDocumentModel,buildHstDocumentModel,buildInter
faces,buildJSky,buildOCM,buildPDFTool,buildSEA,buildSubmissionClient,buildSubmissionDa
emon,buildTextTool,buildToolInterface,buildUtilities,buildVisitPlanner,buildVTT"
   />

   <target name="redeliver"
           depends="cleanall,deliver"
   />

   <target name="javadoc">
      <mkdir dir="${basedir}/API"/>
      <delete includeEmptyDirs="true">
         <fileset dir="${basedir}/API"
                  excludes="CVS"/>
      </delete>
      <mkdir dir="${basedir}/API"/>
      <javadoc maxmemory="128m"
               Private="true"
               destdir="${basedir}/API">
         <packageset dir="${basedir}/APT"/>
         <packageset dir="${basedir}/APTIDServer"/>
         <packageset dir="${basedir}/DocumentModel"/>
         <packageset dir="${basedir}/HstDocumentModel"/>
         <packageset dir="${basedir}/Interfaces"/>
         <packageset dir="${basedir}/JSky"/>
         <packageset dir="${basedir}/OCM"/>
         <packageset dir="${basedir}/PDFTool"/>
         <packageset dir="${basedir}/SEA"/>
         <packageset dir="${basedir}/SubmissionClient"/>
         <packageset dir="${basedir}/SubmissionDaemon"/>
         <packageset dir="${basedir}/TextTool"/>
         <packageset dir="${basedir}/ToolInterface"/>
         <packageset dir="${basedir}/Utilities"/>
         <packageset dir="${basedir}/VisitPlanner"/>
         <packageset dir="${basedir}/VTT"/>
      </javadoc>
   </target>
```

Installing and Using ANT

- **Windows**
  - Download Version 1.5 from**:**

http://jakarta.apache.org/builds/jakarta-ant/release/v1.5/bin/

  - To run, you must have the environment variables **ANT_HOME** (pointing to your *ANT* installation) and **JAVA_HOME** (pointing to your favorite JDK installation) defined.
  - The *ANT* manual recommends installing it in a short named place like **c:\ant**.
  - Set an alias to **%ANT_HOME%/bin/ant.bat** if you like.
- **Solaris**
  - *ANT* 1.5 is already installed. Add the **ANT** package to your .envrc file and define **JAVA_HOME.**
  - Set an alias to **${ANT_HOME}/bin/ant** if you like.
- **Mac OS X**
  - Well, you guys can figure it out.

Plans for Overnight Builds

- **Main Branch Builds** (start this week)
  - **Tasks**
    - **BuildNoFail** performed to check that each sub-system compiles against the committed libraries.
    - **JavaDoc** is generated from committed code.
  - **Build Failures**
    - Detected by grepping for “BUILD FAILURE” in the *ANT* output.
    - Email sent every night if there is a failure and only when there is a failure.

The APT nightly build failed:

The log file is /project/ra5/apt/CM/nightly-build/ant/Mon/build.log.
The build world is /project/ra5/apt/CM/nightly-build/ant/Mon with tag APT_Tue_ant_build.

```
    [exec]     [javac] /project/ra5/apt/CM/nightly-
build/ant/Mon/APT/APT/edu/stsci/apt/controller/AptController.java:15: 'class' or 'interface'
expected
    [exec]     [javac] dpackage edu.stsci.apt.controller;
    [exec]     [javac] ^
    [exec]     [javac] /project/ra5/apt/CM/nightly-
build/ant/Mon/APT/APT/edu/stsci/apt/controller/AptController.java:17: 'class' or 'interface'
expected
    [exec]     [javac] import edu.stsci.tina.controller.DefaultTinaController;
    [exec]     [javac] ^
    [exec]     [javac] 11 errors

    [exec] BUILD FAILED
    [exec] file:/project/ra5/apt/CM/nightly-build/ant/Mon/APT/APT/build.xml:74: Compile
failed; see the compiler error output for details.

    [exec] Total time: 8 seconds
```

[exec] Result: 1

BUILD SUCCESSFUL
Total time: 2 minutes 18 seconds

- **Build Archive**
  - The last week's build worlds will be maintained on aura.stsci.edu under /project/ra5/apt/CM/nightly-build
  - Links to last three successful builds will be kept.
    - **earliest:** the earliest of the last three good builds.
    - **previous:** the second most recent good build.
    - **latest:** the most recent good build.
  - Link to JavaDoc of latest good build will be maintained.

- **Delivery Branch** (start in about a month)
  - **Redeliver** performed on latest delivery branch to insure the system builds properly from scratch.
  - **JavaDoc** generated from the committed code.

- **InstallAnywhere** (start in about a month)
  - Installers maintained for last three successful overnight builds and overnight deliveries.
  - May even automatically be installed on Solaris clusters.

And in the End

- Starting with **APT/README.build**, there are lots of comments in all the relevant files.
- Once ANT is installed, anyone should be able to
  - Build APT on any of our development platforms

    (Invoke "**ant**" from the APT top level directory)
  - Add a sub-system or change dependencies in the build process

    (modify the file APT/AntBuildData.xml)
- If any of this stuff doesn't makes sense, feel free to ask for help.