

Importance of Build System Inspection by IV&V

Build systems, like the software itself, must be maintained during the lifetime of a project

- **Unnecessary code files may be compiled and included**
 - **Test code could be compiled into the flight image which could cause issues with free-space requirements**
 - **Files that are only used in a lab environment may make it into flight code, which will be tested under the assumption this code is not present**
- **Longer new developer setup times**
 - **More time to work out issues with their build environment before they are able to contribute to the project due to differences in the way dependency paths are handled**
 - **Lack of configurable paths to dependencies may require new developers to copy existing files, wasting space on their machines**
- **Adding, removing, and moving configurations and tests can become cumbersome**
 - **If the build system is not laid out in a hierarchical way, intra-project include paths and dependencies can become tangled and require significant work to change**
 - **Adding a new target may require files from many different places of the overall project, so a lot of duplicate configuration or hunting for paths may be required to continue on with work**

Build systems should make working with and expanding a project easy. Their configuration files should be powerful, yet intuitive. If care is not taken when constructing them, including a solid foundation for design up front, they can quickly become a mess. Scripts that set up the environment for build systems should be developed for and tested on all host platforms that the developers intend the project to be compiled on.

Sometimes the build system can be “damaged” by people handling the files along the way. For example, if a project that is intended to be built on a Linux system is checked out of a repository on a Windows machine, the line endings can be altered. This can be mitigated by setting up the repository in such a way as to not do this, or by an engineer who is going to build the software check some files for line endings first. This is a problem that IV&V can often cause because of how we obtain software from developers.

GPM 2.1 was the first piece of project software that the ITC team built. We found the following issues:

- The “setvars.sh” file had MS-DOS style line endings despite the project being intended to build in a Linux environment. This caused cryptic errors during building because of how the environment variables are set up. What should have been
`/home/gosim/gpm21/build/mission_inc` was
`/build/mission_inc/` because the first part of the string was over-written due to the carriage return in the file. In fact, the entire project has DOS style line endings.
- Most parts of the project correctly included vxWorks in their path with the \$WIND_BASE environment variable. However, some parts (psp, cdh_lib, xb, sw, tm) used a hard-coded path to `/opt/WindRiver/vxworks-6.4/`. This requires a developer to set up a symlink or copy files on his machine due to this lack of consistency. This is also a maintenance issue if the WindRiver directory ever needs to be moved.
- `cdh_lib` cannot find `LRO_xbdRamDisk.h` in a WindRiver directory (with LRO BSP) because the paths are set up incorrectly. (It’s also possible the BSP is not installed in the correct place.)