Validation_Approval Documentation:

Main Program:

This program is taking advantage of Class OwnerDependencyCheck to verify the file changes.

Arguments:

1. **CircularDependency (-C)**
    a. Prevent Circular Dependency [Default='True']
    This flag will verify the dependencies, and terminate if there are any Circular dependencies which cannot be resolved
2. **Directory* (-D)**
    a. Directory to scan
    b. The Directory path to parse
3. **Approvers* (-A)**
    a. Comma Separated Signatures [Default='']
    b. The Signatures separated by comma
4. **ChangedFile* (-Ch)**
    a. Path to file to verify
    b. If the path is not valid, it will loop for the parents and pickup parents properties
5. **Multiple (-M)**
    a. Multiple Verification [Default='False']
    b. This will allow the user to verify multiple files one by one

Example:

Validation_Approval.exe -D C:\Users\Test1 –A A,C -Ch C:\Users \Test1\Dir1\file1

Known Issues:

None.

OwnerDependencyCheck Documentation:

This Library uses QuickGraph to store all the dependencies. And verify if there are enough approvals for the changed/ added/removed files. It loops through all dependencies and verify that at least one of the Owens signed the change.

Public Methods

```
public OwnerDependency(String Path = null, bool AvoidCircularDependenc
y = false)
```

1: Path to parse:

This is the root path to parse all dependencies.

This is an optional argument to speed up coding and initialization, for simple solutions.

Note: For more complex solutions (multiple island of folders), use AddPath(string Path)

2: Circular Dependency:

This is an optional check, to verify the scanned directories do not have any Circular Dependencies, and throw exception (CircularDependencyDetected). This option could cause extra time while scanning the files and folders, since it needs to check for all dependencies related (find a path to dependences), yet it will prevents infinite loops and dependencies. By disabling the option, you may have slightly faster parsing the directories.

```
public void AddPath(string Path)
```

Add path to parse. The path needs to be valid

```
public bool Verify(string file, string Signed = null)
```

Verify the file based on the people who signed this action. Those files does not need to exists at the time of scan (adding file), all permissions and dependencies will be pick from the parents of the folder. This will allow the user to add or remove files without scanning

Additional features are added, due to ensuring quality of work:

1: Checking to Circular Dependency. Since (as it mentioned) there might be a huge file system, protecting circular dependencies is a need.

2: Not all files needs to be there at the time of the scan. Sometimes the user just added a file after scanning or some folders do not have any owner or dependencies.

3: Multiple directory roots are supported. It is very relevant to have multiple directories where they are not all under the same tree, This software can handle multiple directory roots.

This software will go up to directors (to their parents) to find the most relevant Owers and dependencies.

Issues :

Nothing known

TODO:

There are some room to improve CircularDependenc check efficiency .