Subject: Ultimate++ i.r.t. Fossil SCM
Posted by alkema_jm on Mon, 18 May 2020 16:15:53 GMT
View Forum Message <> Reply to Message

LS,

I look how I can change window in the next template. I think that these attributes are 'better' in relation to revision control systems:

First I look to UMK.exe, specially the Makefile :)

In revision control systems, a repository[1] is a data structure which stores metadata for a set of files or directory structure. Depending on whether the version control system in use is distributed (for instance, Git or Mercurial) or centralized (Subversion or Perforce, for example), the whole set of information in the repository may be duplicated on every user's system or may be maintained on a single server. Some of the metadata that a repository contains includes, among other things:
• A historical record of changes in the repository.
• A set of commit objects.
• A set of references to commit objects, called heads.
https://en.wikipedia.org/wiki/Repository_(version_control)

> This example is from 2009, probably did not matured well with time...

I will try to put a phase field in selection screen:
Development, Testing, Acceptance and Production (DTAP)[1][2] is a phased approach to software testing and deployment. The four letters in DTAP denote the following common steps:
1. The program or component is developed on a Development system. This development environment might have no testing capabilities.
2. Once the software developer thinks it is ready, the product is copied to a Test environment, to verify it works as expected. This test environment is supposedly standardized and in close alignment with the target environment.
3. If the test is successful, the product is copied to an Acceptance test environment. During the Acceptance test, the customer will test the product in this environment to verify whether it meets their expectations.
4. If the customer accepts the product, it is deployed to a Production environment, making it available to all users of the system.
 https://en.wikipedia.org/wiki/Development,_testing,_acceptan ce_and_production

Greetings Jan Marco

Appendix A: Info from internet:

About Fossil SCM (Source Control Management)
Fossil is an open-source version control system: it's fully based on SQLite, and it comes from the same author of SQlite (Dr. R. Hipps).
More or less, Fossil is the same as the best known GIT or SVN; but Fossil is unbelively simple

and lighweight compared to them.
That's not all: Fossil supports many features typical of web-based project management and bug-tracking tools such as Trac or RedMine.
See also http://www.gaia-gis.it/gaia-sins/about-fossil.html

File Attachments
1) IDE_Main_repository_01.jpg, downloaded 757 times

Subject: Re: Ultimate++ i.r.t. Fossil SCM
Posted by alkema_jm on Wed, 20 May 2020 07:02:00 GMT
View Forum Message <> Reply to Message

LS,

//UMK - Command line builder

ist :umk assembly package [build_method] [-[a][b][u][r][s][S][v][m][d][M][M=makefile][l][x][X][Hn]].. [+FLAG[,FLAG]..] [out] [! [runarg]..]

soll:umk package repository environment command [build_method] [-[a][b][u][r][s][S][v][m][d][M][M=makefile][l][x][X][Hn]].. [+FLAG[,FLAG]..] [out] [! [runarg]..]

package = {"uppsrc", "myapps", "libreoffice", "tutorial", "bazaar"}
repository = {"umk.fossil", "ide.fossil"}
environment = {"development" or "d", "testing" of "t", "acceptance" or "a", "production" or "p"}
command = {"makefile", "runmakefile", "makedatabase", "runmakedatabase", "cmake", "ninja", "gn", "init", "open", "close", "add", "rm", "addremove", "push", "pull", "sync", "commit", "clone", "ui", "fossilserver", "torlistener", "compile", "link", "compilelink", "install", "installexe"}
build_method = {"MSVS17.bm", "MSVS17x64.bm"}

I will try to integrate main from umake.cpp with the main in Fossil SCM.
If I make a repository. All files used for compiling/linking of umk will be put in repository with the "internal" Add-Fossil-command integrated in the UMK source code.
Only after compile/link is succesful, then the Commit-Fossil-command will actually put the files in the umk.fossil repository.

Greetings Jan Marco

Appendix A: Screen current implementation:

File Attachments
1) umk_with_Fossil_SCM.jpg, downloaded 727 times

Subject: Re: Ultimate++ i.r.t. Fossil SCM
Posted by alkema_jm on Thu, 28 May 2020 20:41:55 GMT
View Forum Message <> Reply to Message

Hello,

First issue is that parsing the "the source" files for "include files" is implemented in "ide" part only, not in the "umk" part.

I try to put the ("Ide") include files in Mysql:

Now I will search for the files needed for compilation/linking.

Goal is to determine which files must to added to the Fossil file "ide.fossil" to let the repository compile/link.

Greetings Jan Marco

File Attachments

1) ultimate_filepathInclude_01.jpg, downloaded 612 times

Subject: Re: Ultimate++ i.r.t. Fossil SCM
Posted by alkema_jm on Sat, 06 Jun 2020 08:01:29 GMT
View Forum Message <> Reply to Message

Hello,

I feel me an newby in the Ultimate++ habitat. Code is very mature and complex. I must study a lot how it works.
I am logging at different places the include file dependencies to MySQL. Ide environment could be work other than umk environment.

In the Ide environment I see 2 calls to AddDependency(..) when I compile Ide-environment:

```
class Hdepend {
 struct Info {
  Time                time;
  Vector<int>         depend;
  Vector<bool>        bydefine;
  Index<String>       macroinclude;
  Vector<String>      define;
  bool                flag;
```

```
 bool            macroflag;
 bool            timedirty;
 bool            guarded;
 bool            blitzprohibit;
};
```

```
 void   Include(int line, char *filenaam, char *functie, const char *trm, Info& info, const String& filedir, bool bydefine, const String& parent_path, unsigned long depth);
 void   ScanFile(int line, char *filenaam, char *functie, const String& path, int map_index, const String& parent_path, unsigned long depth);
 int    File(int line, char *filenaam, char *functie, const String& path, const String& parent_path, unsigned long depth);
 void   AddDependency(int line, char *filenaam, char *functie, const String& file, const String& depends);
};
```

I will look beter in Hdepend class and prepere Fossil to the Umk main environment,

Greetings Jan Marco

File Attachments

1) add_dependencies_records.jpg, downloaded 620 times