SyWiCo is a tool for synchronizing files
between unconnected machines.

v0.2 © Bruno Carle, August/July 2007

AOverview

- SyWiCo is a tool for managing concurrent modifications of shared files between unconnected computers.

- Its main design goal was for **synchronizing the source code by email between developers, when the use of a source control software is not possible.**

- In order to make the synchronization possible without connection, the communication is stateless and reduced to a very small number of messages - typically 2 or 4.

- This tool is not intrusive, i.e. it does not create any files in the directories that it synchronizes.

- Furthermore this tool **is extremely simple to use for one of the participants, as it leaves one participant free from dealing with merging and conflict resolution**.

- Please note that this is not a tool to replace source control programs, as it provides no versioning.

BDescription of the process

B.1Participants

The two participants sharing the data to be synchronized are called the company and the developer in this document.

The participants have different roles:

- One of the participant is called the '**company'**. On the company machine the changes are applied only when they are ready.

  There is only one directory used by SyWiCo on the company machine.
  It is refered to as '**company work'**. It contains the files that will be synchronized with SyWiCo and can be modified directly to incorporate changes from the company.

- The other participants are called the '**developer'**. All the merging and conflict resolution will be done on the developer machine.

  On the developer machine two directories exists:
  - The **work**:On this directory the developer perform its changes.
  - The **base**: it is an image of the files on the company machine. This directory is managed by SyWiCo and should never be modified manually.

B.2Synchronisation

Basically synchronisation occurs as follows:

- The developer submit his/her changes

- The company process the request, either applying the changes or rejecting them. In both case an update is sent for the developer

- The developer applies the update. If changes occurred both on the developer and company side, they will be merged and the developer should review the changes, fix the conflicts if any, and then resubmit the changes.

A message exchange is always done in this same way. Following is a description in more details.

B.2.1Developer creates and send Submit message

The developer compiles its list of changes of its work directory against the base, which is an image of the latest known copy of the company.
The developer send a Submit message, which includes a checksum of all files in the base in addition to the changes of the developer.

B.2.2Company processes Submit and reply with Update

The company receives the Submit message, and check if the checksum of the developer base is same as the company work directory. Two cases can occur:

- If checksums are the same, the changes of the developer are applied, and the company sends an Update message that contains the updated file.

- If the checksums differs, the company sends in an Update message the files that should be updated in the developer directories.

Note that in both cases an Update message is sent back to the developer.

B.2.3Developer processes Update

The developer receives the Update message, and applies the changes in the base directory and merge the changes in the developer work directory
At this point two cases can happen:

- The base and work directory are the same, meaning that the developer and company are synchronised

- The base and work and different, meaning that the company changes have been merged on the developer machine, so the developer still should check the merge, solve conflict if any, and then resubmit his/her changes to the company with another message exchange.

B.3Length of the synchronisation process

If only one participant performed changes, the synchronisation will require one message exchange (i.e. 2 messages).

If both participants performed changes it will require two messages exchange (i.e. 4 messages)

In case changes are performed in the middle of one message exchange, it may require an additional message exchange

CGUI short manual

This is a short introduction to let you use the SyWiCo gui without reading all the manual – who would do that anyway?

C.1Setup

First of all choose your side: One side is the company, the other is the developer. There can be only one company but several developers. All the synchronisation job will be done on the developers side.

C.1.1Prepare your files

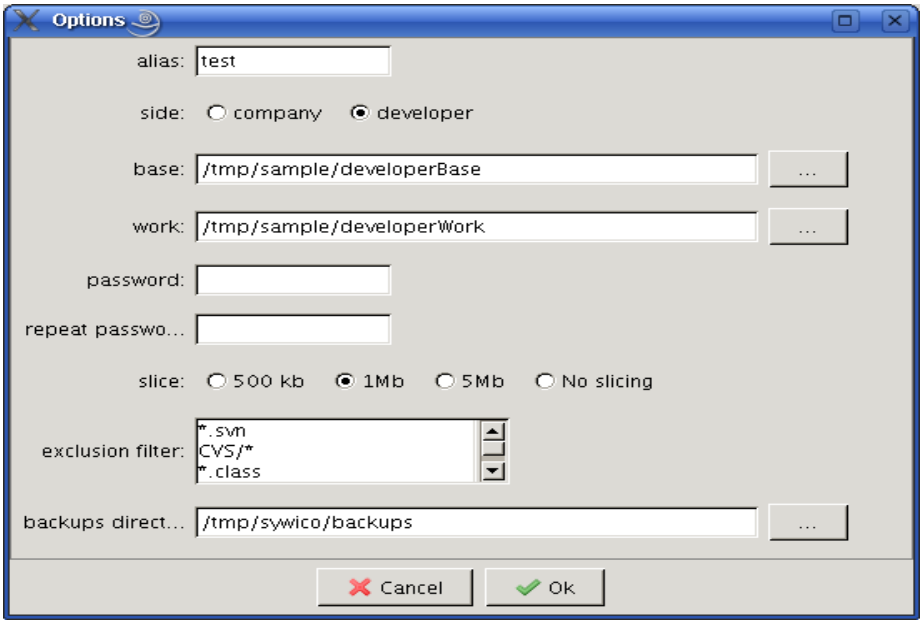
On the company side no file preparation is required. On the developer side, obtain an exact copy of the company files, and put it in a directory that we will refer to as the base. After the setup, the contents of the base directory should never be modified manually.

C.1.2Setup sywico GUI

To launch the GUI, click on the sywico-xxx-gui.jar file, or type in a command line:

```
java -jar sywico-0.2-gui.jar
```

On both sides, setup the project by clicking on Menu>Setup



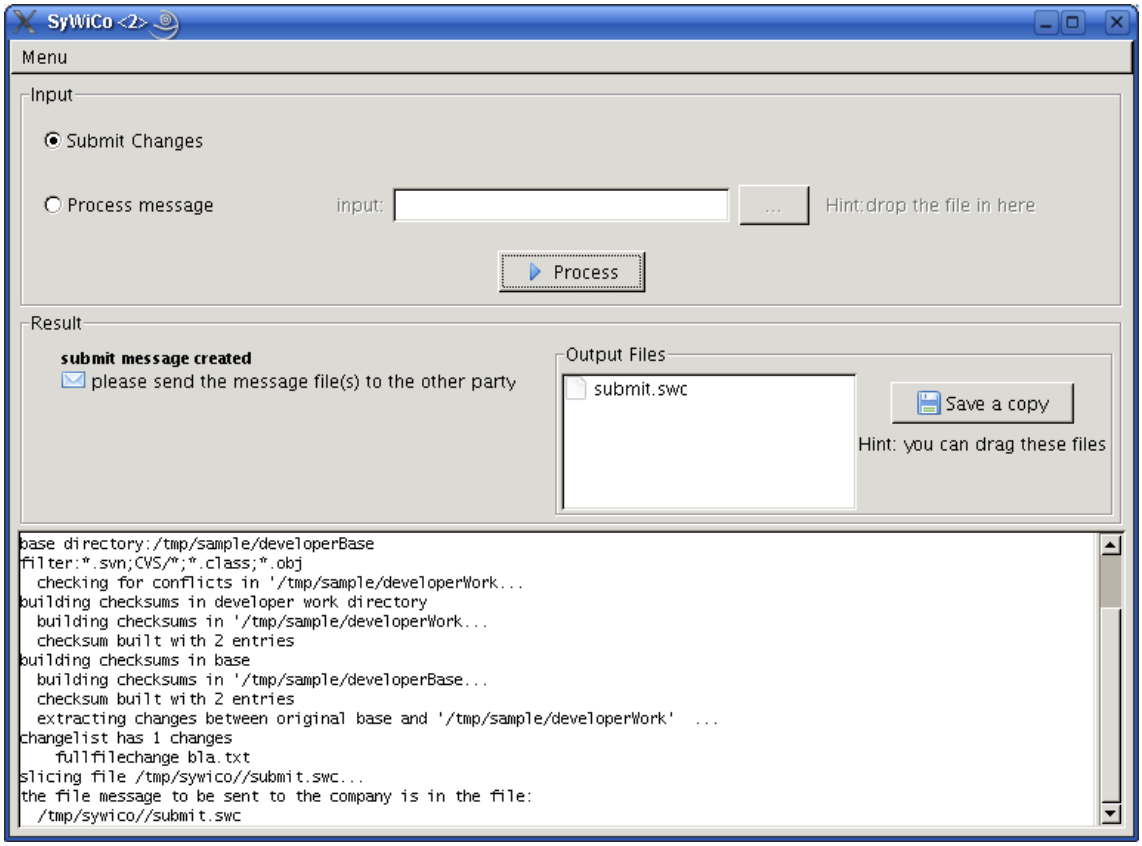
That s all for the setup

C.1.3Synchronisation Message exchange

C.1.3.1 Developer submit changes

Every time you want to synchronise your files, it is always the developer who will send the first message.

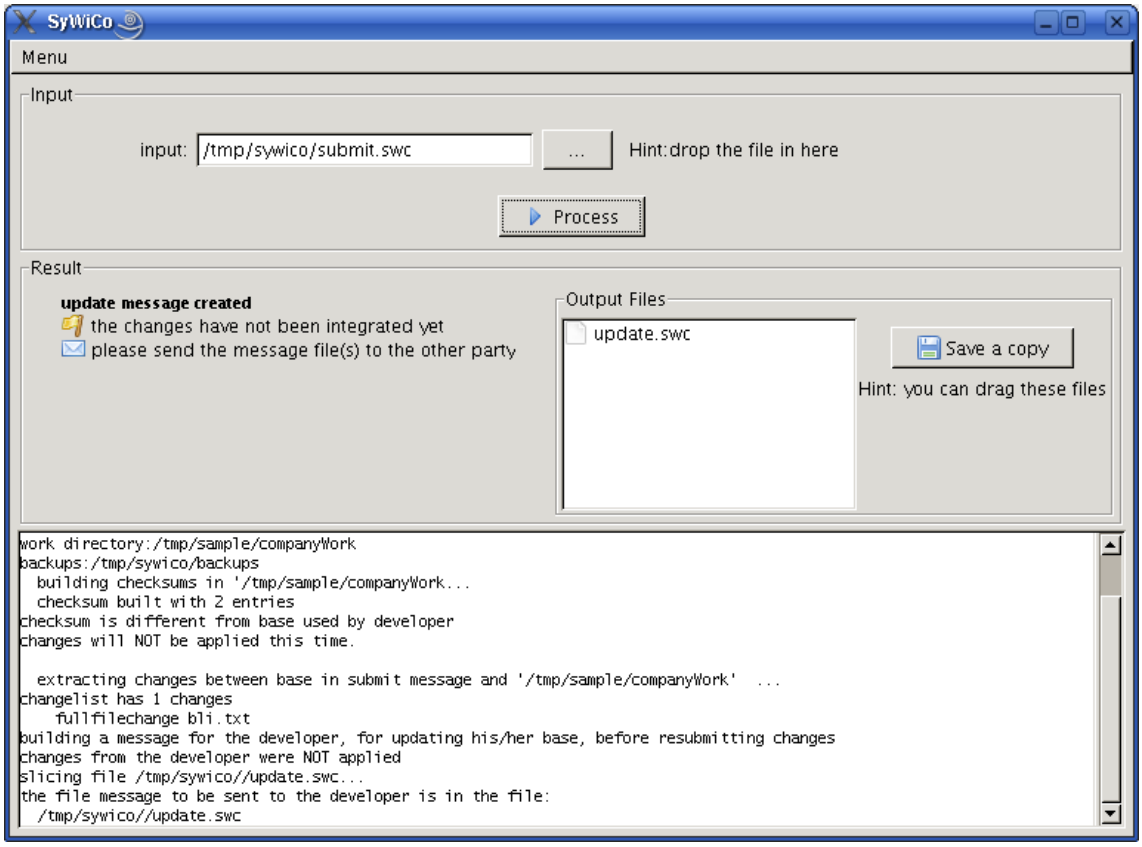
So on the developer side, select the Submit Changes radio in the input panel, and click on process. When the operation is done, you should see something like :



Send the output file (submit.swc) to the company by email, usb stick, whichever medium you want.

C.1.3.2Company processes changes

Once the message arrive on the company side, locate the message file or drag it into the input panel, then click process (notice that the input panel is different on the company side). Once the progress bar completes, SyWiCo will look like that:



At this point SyWiCo will basically indicate either :

- The changes performed by the developer were integrated

or

- The changes were not integrated coz the developer needs first to integrate changes from the company (shown in the picture above)

Whichever is the outcome, you should send the message file (update.swc) to the developer.

C.1.3.3 Developer receives update

Back on the developer machine, specify the message file sent by the company in the input panel and press process. When the process completes, the outcome will be one of the following:

- The company and developer files are synchronized
- The developer files and company have been merged. The changes should be reviewed and then resubmitted to the company (i.e. return to step 1)
- The developer changes and company changes could not be merged, because of some conflicts (e.g. simultaneous modification of binary files). The conflicting files should be manually solved, and then remove the .swc files to indicate that the conflicts are gone. Eventually resubmit the changes to the company.

For more about solving conflicts, see paragraph E.

DDetailed and Command Line Usage

D.1Setup

- Download SyWyCo on both the company and developer(s) machines... (Do I really need to say that?)

- Make sure you that the files composing the project are exactly the same on the company and developer participant.

- On the developer participant, you will need to make a copy of the project, called the ‘base’. This copy will be used to reflect the project files on the company participant.

That s all for the setup.

D.2Synchronizing

Synchronizing is very simple. There are quite a few details in this sections so it may appear more complicated than it is really.

D.3Build the submit message

On the developer machine, type (without the newlines):

```
01  java -jar sywico-0.2-nodeps.jar
02  create_submit_msg
03  -out submit.msg
04  -alias test
05  -dwork sample/developerWork
06  -dbase sample/developerBase
07  -filter "*.svn;*.class;bin"
08  -pass pass
```

Here is an explanation of the lines :

Line	Comment
01	Launch the sywico tool using java.
02	Tell sywico that we want to create a submit message
03	The name of the file that will contain the message. If the file already exist it will be overwritten
04	An alias for the project – not currently used, but still you have to put a value
05	The developer working directory
06	The base directory. This is a copy of the company data
07	Filters used to ignore some files. Several filters can be specified, separated by a semicolon
08	Password used to encrypt the message

You should get an output like :

```
01    cmd:create_submit_msg
02    output file:submit.msg
03    alias:test
04    working directory:sample/developerWork
05    filterExpressions:.*\.svn
06    base:sample/developerBase
07
08  create a submit message
09  building checksums in developer work directory
10    building checksums in 'sample/developerWork' ...
11    checksum built with 2 entries
12  building checksums in base
      building checksums in 'sample/developerBase' ...
      checksum built with 2 entries
      extracting changes between original base and 'sample/developerWork'  ...
      changelist has 1 changes
        update ./bla.txt
    the message to be sent to the company is in: submit.msg
```

At this point, the developer opens his favorite email client and sends the submit.msg file to the company. Alternatively any medium can be used – a dog carrying a floppy will work too, provided they are enough sausages on the way.

D.4Process the Submit, and create an Update message

On the company machine, type:

```
01  java -jar sywico-0.2-nodeps.jar
02  process_submit_msg
03  -in submit.msg
04  -out update.msg
05  -cwork sample/companyWork
06  -backup sample/backups
07  -pass pass
```

And again here is the explanation :

Line	Comment
02	Tell sywico that we want to process a submit message. This will also create an update message
03	The name of the file that contains the Submit message to be processed.
04	The name of the file that will contain the update message.If the file already exist it will be overwritten
05	The company working directory
06	An optional directory were the backups should be copied. If this parameter is not specified, then backups are skipped… and rollback is not possible in case of error.
07	Password used to decrypt and encrypt the messages

This time the output will be something like :

```
01  welcome to SyWiCo v0.1!
02    cmd:process_submit_msg
03    input file:submit.msg
04    output file:update.msg
05    working directory:sample/companyWork
06    backup directory:backups
07  opening message file submit.msg...
08
09  processing a submit changes message from the developer
10  building checksum in company directory
11    building checksums in 'sample/companyWork' ...
12    checksum built with 2 entries
13  checksum is same as base used by developer
14  performing a backup copy of sample/companyWork in backups/2007-07-01-14h25m08/companyWork...
    changes to apply:
15    changelist has 1 changes
16      update ./bla.txt
17  applying changes...
18  changes applied!
19
20  building a message for the developer, for updating his/her base directory
21    building checksums in 'sample/companyWork' ...
22    checksum built with 2 entries
23    extracting changes between original base and 'sample/companyWork'  ...
24    changelist has 1 changes
25      update ./bla.txt
26  the message to be sent back to the developer is in: update.msg
27
28
```

However, if some changes were performed on the company participant, the output would be different, and the lines 09-28 above would be replaced by :

```
09    processing a submit changes message from the developer
10    building checksum in company directory
11      building checksums in 'sample/companyWork' ...
12      checksum built with 2 entries
13    checksum is different from base used by developer
14    changes will NOT be applied this time.
15
16      extracting changes between base in submit message and 'sample/companyWork'  ...
17      changelist has 1 changes
18        update ./bli.txt
19    building a message for the developer, for updating his/her base, before resubmitting changes
      the message to be sent back to the developer is in: update.msg
20
```

In both cases the Update message should be sent back to the developer

D.5Developer participant again

We are almost done ! The developer will receive the update message and process it :

```
01      java -jar sywico-0.2-nodeps.jar
02      process_update_msg
03      -in update.msg
04      -dwork sample/developerWork
05      -dbase sample/developerBase
06      -backup sample/backups
07      -pass pass
```

D.5.1Synchronized !

Now, if there were no changes in the company participant, the output will be

```
01        cmd:process_update_msg
02        input file:update.msg
03        working directory:sample/developerWork
04        base:sample/developerBase
05        backup directory:backups
06      opening message file update.msg...
07
08      processing a update message sent by the company
09      checking base...
10        building checksums in 'sample/developerBase' ...
11        checksum built with 2 entries
12      performing a backup copy of sample/developerBase
13      performing a backup copy of sample/developerWork
14      building list of current changes made by the developer...
15        building checksums in 'sample/developerWork' ...
16        checksum built with 2 entries
17        extracting changes between between 'sample/developerBase' and 'sample/developerWork'  ...
          changelist has 1 changes
18          update ./bla.txt
19      applying changes sent by the company...
20      rebuilding list of changes sent by the company...
21        building checksums in 'sample/developerBase' ...
22        checksum built with 2 entries
23        extracting changes between between 'sample/developerWork' and 'sample/developerBase'  ...
24        changelist has 1 changes
            update ./bla.txt
25      merging changes from the company and the developer ...
26        building checksums in 'sample/developerWork' ...
27        checksum built with 2 entries
28      the developer changes have been applied on the company participant
29      the company files are now synchronized
30
31
```

At this point the working directory, and base directory on the developer machine are the same, and are also the same as the working directory on the company machine.
That 's it ! unless there were some changes in the company…

D.5.2Ups not yet...

However if there were some changes in the company, the lines 27-31 of the output will be replaced by

```
27          Merging changes from the company and the developer ...
28          building checksums in 'sample/developerWork' ...
29          checksum built with 2 entries
30        a merge occured. please check and resubmit your changes
```

And, in case of conflicts, the output would be:

```
27          Merging changes from the company and the developer ...
28          building checksums in 'sample/developerWork' ...
29          checksum built with 5 entries
30      alas 1 conflict(s)  found. please fix them, and then resubmit your changes
```

In this last two cases, the developer should review the merged files and fix the possible conflicts. For information on conflict resolution see next section.
After that he/she should resubmit the changes again, as described above.

EHandling conflicts

E.1Different versions of the conflicting file

In case of conflict, SyWiCo will create at least one the following versions of the file.
For example for a conflicting file named bla.txt, the copies created could be:

File	The file represents
Bla.sw-base.txt	The file as it was in the base directory before the merge.
Bla.sw-developer.txt	The file as it was in the developer directory before the merge.
Bla.sw-company.txt	The file as it is in the company. (which is same as in the current base)

In addition, if the conflicting file is binary it is deleted, and only the copies remain.
However, if the conflicting file is a text file, the file will be maintained and the conflict will appear inside the file, as described next.

E.2Conflicts inside text files

The conflicts are shown in the text files with some special marker lines.
Let s use a concrete example for that. Imagine that the file bla.txt is as follows when the developer and company last synchronized:

```
01      one
02      two
03      three
04      four
05      five
06      six
07      seven
08      eight
09      nine
```

Now imagine that the developer changed the line 4 by “hello from developer”, And that the company deleted the line 4 and changed line 5 by “hello from company”

The merged conflicting file will be:

```
01      one
02      two
03      three
04    --# conflict on lines [4,6[ # --
05      --# Company side #--
06      --C*hello from company
07      --#  Developper side # --
08      --D*hello from developer
09      --D five
10      --# end of conflict –
11      six
12      seven
13      eight
14      nine
```

The developer has to edit the file to fix the conflict.

Here is the list of all meaning of actual markers

Marker	Meaning
--S	Start of the conflict
--E	End of the conflict
--#	Comment
--C	Unchanged line on the company side
--C*	Changed line in the company
--D	Unchanged line on the developer side
--D*	Changed line on the developer side

E.3Conflicts between directories

In some case conflicts can occur involving directories and files altogether. In this case the original file or directory will be replaced by two copies: one holding the company changes and one for the developers. The name used are as described in the paragraph E.1 .

E.4Marking a conflict as solved

Once the conflict has been manually fixed by the developer, he/she should removes the various versions of the conflicting file.

Otherwise the Submit message will not be created.

FSlicing the messages

The messages files can be sliced. This is controlled by the option '–split' .
By default the messages are split in slices of 1Mb.

Note that when specifying an input file ('-in' parameter) the regular file name should be used, not the one of the slices.

SyWiCo will first reassemble the slices into the regular file, and then attempt to process the message.

A special command is provided to reslice a message.
For example to reslice a message in slices of 500k you can use:

```
01  java -jar sywico-0.2-nodeps.jar
02  reslice_msg
03  -file message.msg
04  -split 500
```

This command will work whether the message.msg file is already sliced or not.
If you just want to reassemble a file (aka to unslice) use the parameter value:
–split 0

GAdditional considerations

G.1Obtaining the first copy from the company

To obtain the first developer copy from the company, create empty base and work directories, and send an submit message.

G.2Requesting an update without submitting changes

It is possible to request an update from the company side, without actually submitting any changes. Simply use the base directory instead of the work directory when building the submit message (line 5):

```
01  java -jar sywico-0.2-nodeps.jar      \
02              create_submit_msg \
03              submit.msg \
04              test \
05              sample/developerBase \
06              sample/developerBase \
07              ".*\\.svn"
08
```

G.3Newlines

The merge can work only on text files, in which lines are separated by … newlines!
For this reason, it is important that the files should use the newlines as they are handled by default on the developer machine.
TODO: put an option to change that

G.4Security

TODO: make sure that SyWiCo will never read nor modify files outside of work and base directories

G.5Known bugs

On linux the GUI does not support Dnd.

G.6Disclaimer

This tool comes with ABSOLUTELY NO WARRANTY. Sywico is free software, and you are welcome to redistribute it under certain conditions. See the licence.txt file for more details.