Chapter

4 *Organizing a Project*

This chapter explains how diagrams and other documents that can be handled in the Organizer, may be managed in a project including several project members. The functionality of the SDL suite related to management of diagrams in a project environment is provided by the Organizer. A reference manual for the Organizer can be found in chapter 2, *The Organizer, in the User's Manual*.

Introduction

General

Software development projects are typically staffed by a large number of software engineers, each working simultaneously on different parts of the same system. This requires careful coordination and version control of the different parts of the system. Normally, stable (well-defined) versions of a system are stored in a central storage area (in this chapter named the *original area*) accessible for each user. The original area is typically either:

- A directory containing different versions of the system managed by a version or revision control system, e.g. RCS, CM Synergy or ClearCase.
- An ordered set of directories, where each directory represents one version of the system and contains the information that has changed as compared to the previous version. For example, version 1 of the system PBX may be stored in the directory PBX-1 which contains version 1 of all parts of the system PBX. Version 2 of the system PBX is then stored in the directory PBX-2 which contains only the parts of the system that has changed in comparison to PBX-1.

In addition, each project member may have his own *work area* where he temporarily stores the parts of the system he is currently working on. Furthermore, a project may also use a reference area which contains one or more packages of information that are shared between different projects.

This way of working is supported by the Organizer tool. It allows the members of a project to work independently, but in a coordinated and structured way. In particular, the tool supports:

- Individual customization of the user environment.
- Easy visualization of the SDL system's contents, in terms of SDL packages and diagrams, Message Sequence Charts and physical files.
- Updating of versions from each individual user's work area to the shared original area.

- Copying of SDL diagrams and Message Sequence Charts between different storage areas.

Improved Support

To facilitate the management of more complex projects, the SDL suite can be integrated to control systems that are used to manage versions and revisions. Example of such systems are RCS, CM Synergy and ClearCase. The Organizer can easily be customized in order to extend the menus commands with support for “check-in”, “check-out” and “update” operations. The extensions are defined in textual files that contain the definitions of the names of the menus and commands, what operations they will call in the control system, what objects are the subject of the operation, etc.

The rest of this chapter is organized as follows:

- “Diagram Binding” on page 152 describes the principles for how diagrams and other documents are connected to files.
- “How to Manage the Diagrams in a Project” on page 155 describes how the tools can be used to manage the SDL diagrams in a project environment.

Diagram Binding

The Organizer provides the ability to manage the diagrams and other documents that build up a project, without the need for an external version or revision control system.

Important is that one diagram is represented by one file, and the mapping between diagrams and files are kept in the *system file*. The system file is the file that is opened when issuing the *Open* command in the Organizer. The system file has by default the extension `.sdt`.

A diagram may be bound to a physical file using the Organizer by:

- Automatic Binding
- Manual Binding

Automatic Binding

Diagrams may be bound automatically when the diagram is saved for the first time. The name of the physical file will by default be `<diagram name>.<ext>`, where <*diagram name*> is the name of the diagram and <*ext*> is a suggested extension of three letters indicating what type of diagram is contained in the file. See "Save" on page 11 in chapter 1, *User Interface and Basic Operations*, *in the User's Manual* for information on default file names and extensions.

Example 49 ----------

The block diagram with the name DemonBlock will by default be bound to the file `demonblock.sbk`

Manual Binding

Diagrams and files may be bound manually, on demand, via the command *Connect*. A file selection dialog appears and the selected diagram may be connected to an existing file. Another possibility is to look for the diagram file in a certain directory.

Figure 54: Connecting a diagram to an existing file

When connecting a diagram, there is a possibility to automatically connect all diagrams in a possible substructure, the *Expand Substructure* option. The names of the files are not important when mapping files automatically, using this feature, since the diagram name is stored in the file. What is important though, is the extension of the file name which must correspond to the type of diagram stored in the file.

Source and Target Directories

Two settings in the Organizer are related to where diagrams are stored. The two settings are:

- *Source Directory*
  The directory where new diagrams are stored.
- *Target Directory*
  The directory where generated files are stored. Generated files do not include the source files, i.e. the files included in the Organizer Diagram Structure.

The source directory and the target directory may be changed in the Organizer by using the *Set Directories* command. In the *Set Directories* dialog there is also an option to either show the absolute path to files, or to show the path relative the Source directory. For a complete description of the functionality of this command, please see the "Set Directories" on page 70 in chapter 2, *The Organizer*, *in the User's Manual*.

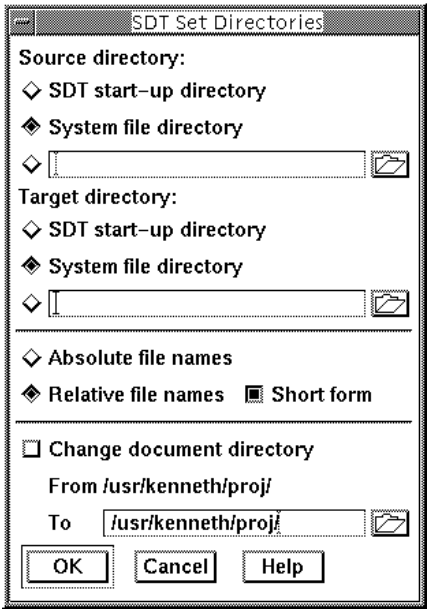

Figure 55: Specifying the Source and Target directories

This affects the way paths to files are stored in the system file:

- If the option *Absolute file names* is chosen, then the absolute path will be stored in the system file.
- If the option *Relative file names* is chosen, then only the relative path will be stored in the system file. This is important if the source directory should be moveable, in which case this option should be selected.

How to Manage the Diagrams in a Project

One way to manage the diagrams in a project is to have the original area managed by some external configuration or version control system, e.g. RCS (Revision Control System), which is available on most computer systems. In that case, every member of the project has his own work area. In each work area, there is a system file (or perhaps multiple system files, each one containing a different part of the system). All diagrams in the system file are bound to files in the users' work area. The relationship between files in the users' work area and files in the original area is managed by the external configuration or version control system.

The rest of this section describes how to use RCS as a version control system. For more information about SDL suite/RCS Integration, see Readme files in `<inst dir>\examples\cm\win32\rcs\` or `<inst dir>/examples/cm/unix/rcs/`. How to use CM Synergy is described in "Using CM Synergy Together with an SDL System" on page 162. How to instead use ClearCase is described in "Using CM Synergy Together with an SDL System" on page 162.

The control is based on the following mechanisms:

- The locking functionality in RCS, by using the check in and check out commands.
- The multi user support in the Organizer, facilitated by the *Control Unit Files.*
- The possibility to add own menu commands to the SDL suite, commands that in this case should connect the Organizer to RCS.

Note:

The following instructions and descriptions only addresses a **UNIX** system with the **UNIX** version of RCS.

Starting to Use RCS Together with an SDL System

When handling of files managed by RCS is to be introduced for an existing SDL system, the following approach can be used:

1. Have all the files related to your system placed in one directory hierarchy which we call the *work area*.
2. Set the *Source Directory* in the Organizer to the work area directory.
3. Now we create the *original area* for the system as the RCS file database. The work area and the original area should have the same directory structure for a given diagram system. The original area will contain the RCS directories that hold the revision files, but these directories are empty for now. (Note that the work area does **not** have any RCS directories.)

Example 50: Work and original area for the DemonGame system––––

The work area has three directories:

- One for the files related to the system.
- One for each of the two blocks (GameBlock and DemonBlock):

```
demonblock      gameblock       system

demonblock:
demon.spr       demonblock.sbk

gameblock:
game.spr        gameblock.sbk   main.spr

system:
demongame.msc   demongame.ssy   systemlevel.msc
```

The original area for the Demon Game system (created in a different place in the file system) will then have the following structure:

```
RCS      demonblock      gameblock       system

demongame/RCS:

demongame/demonblock:
RCS

demongame/demonblock/RCS:
```

```
demongame/gameblock:
RCS

demongame/gameblock/RCS:

demongame/system:
RCS

demongame/system/RCS:
```

4. Define the root directory of the original area (**`setenv rcsroot <directoryname>`**) and load the RCS menu-set into the Organizer (run **`$telelogic/sdt/examples/RCS/sdtrcs.mnu`**). A new menu titled *RCS* appears in the Organizer menu bar.

   – The *RCS* menu in the Organizer is an example of how check in and check out can be done on one diagram and a hierarchy of diagrams. (The commands that are defined in the example could easily be tailored to fit specific needs).

5. Check in the first version of each diagram file, using the Organizer commands *Recursive Check In* or *Check In*.

   – To visualize the results, make sure the file permissions are visible in the Organizer (Choose the *View Options* command in the *View* menu and select the option *File access permissions*.

6. After a successful check in, the original area is now populated with the RCS files.

Example 51: The Original Area, now populated

```
RCS     demonblock      gameblock       system

RCS:

demonblock:
RCS

demonblock/RCS:
demon.spr,v             demonblock.sbk,v

gameblock:
RCS

gameblock/RCS:
game.spr,v      gameblock.sbk,v main.spr,v

system:
```

```
RCS

system/RCS:
demongame.msc,v          demongame.ssy,v
systemlevel.msc,v
```

Using the SDL suite and RCS in a Multi User Environment

If the system is developed by a group of developers it is useful to partition it and assign *Control Unit Files*, for the different partitions. For the DemonGame example, let us assume that the DemonBlock is developed by one developer and the GameBlock by another. The procedure here is to create:

- a top level control unit file,
- a control unit file for the DemonGame system,
- a control unit file for the DemonBlock, and
- a control unit file for the GameBlock.

1. In order to assign a control unit file for an Organizer object we select the object and execute the *Configuration > Group File* command from the *Edit* menu.

2. When the four control units above are assigned, we save the system from the Organizer (which also saves the control units on their respective `.scu` files).

3. Now, we can check in the created `.scu` files. It is wise to have the control unit file located in the same directory as the corresponding object file.

Example 52: The Original Area with Control Unit Files

The DemonGame original area with checked in control unit files:

```
RCS              demonblock       gameblock
system

alfa/RCS:
demongame.scu,v

alfa/demonblock:
RCS

alfa/demonblock/RCS:
DemonBlock.scu,v         demon.spr,v
```

```
demonblock.sbk,v

alfa/gameblock:
RCS

alfa/gameblock/RCS:
GameBlock.scu,v game.spr,v      gameblock.sbk,v
main.spr,v

alfa/system:
RCS

alfa/system/RCS:
DemonGame.scu,v         demongame.ssy,v
demongame.msc,v         systemlevel.msc,v
```

Make Local Changes Global Using RCS

When a user has made changes in files that have been checked out and locked and wants to make these changes global for the complete project, the following steps should be performed.

- Use *Check In* or *Recursive Check In* from the Organizer *RCS* menu, on the diagrams that have been modified. Note that *Recursive check In* will automatically handle control unit files as well.

Example 53: Adding a Diagram to the Control Unit and Saving ––––––

Say that a new process diagram is added to the GameBlock:

1. The user must therefore check out the files `gameblock.sbk` and `GameBlock.scu`.
2. With the SDL Editor, the diagram GameBlock is updated to contain the new process reference symbol.
3. When saving from the Organizer, `GameBlock.scu` will automatically be updated with the new process diagram file name. Note also that all changes will be local to the partition of the diagram system that represents the GameBlock and that no global update of the system file is needed.

Make Global Changes Local Using RCS

If a user is notified that there have been changes to the system, the following should be done in order to update the users' local system:

- Use *Check Out* or *Recursive Check Out* to update the diagrams.
  - From Example 53 above, if the GameBlock has been modified a *Recursive Check Out* on the GameBlock object will bring in the newest checked in version.

Building and Populating a Work Area from RCS based Original Area

Say that we have an original area with its RCS directories containing the checked in diagrams and the control unit files. A developer that is new to the project wants to make a complete update of a system in his work area to get an up-to-date view.

1. The developer must first create/update his work area directory tree. The environment variable `rcsroot` should be defined to point to the original area.
2. In an empty Organizer: the RCS menus are loaded (use for instance the command **`$telelogic/sdt/examples/RCS/sdtrcs.mnu`**), the *Source Directory* is set to the work area, and the top level control unit file is associated with the SDT symbol (using the Organizer *Configuration > Group File* command).
   - Note that the top level control unit file must be assigned the correct file name that is checked in. For the DemonGame example, the top level control unit file name would be `DemonGame.scu` in the work area top directory.
3. To populate/update the work area, select the SDT symbol in the Organizer and perform *Recursive Check Out.*
4. The new developer saves the system file to get a personal view of the diagram system.
   - When saving the system file, the Organizer tries to save the top level control unit file; this file is read-only and the Organizer will hence issue an error message which can be disregarded.

Endpoint Handling with RCS

Presently there is **one** global link file (the *master link file*) that holds the link information for a given system. A group of developers can coordinate changes to this file with the help of the RCS check out and lock facility, where only one developer can update the file at a time. A *local link file* is used for temporary storage of changes made to the endpoint and link information, until the user decides to make his changes global. Say we have checked in the first version of the link file into the original area. In order to modify the link file, the procedure would follow these guidelines:

1. Issue the command *Add Local Link File* in the RCS menu.
2. All changes to the endpoint and link information will now be stored in the local link file, and the master link file will be left unchanged.
3. When it is time to update the global link information, check out the master link file by issuing the command *Check Out And Lock Link File.*
4. Issue the command *Merge Local Link File,* which will update the master link file with the information from the local link file.
5. Check in the master link file by issuing the command *Check In Link File.*

Simultaneous Editing of an SDL Diagram

There is a possibility for several users to work simultaneously on the same SDL diagram by utilizing the commands *Split* and *Join*, available in the *Tools* menu of the Organizer.

By the Split command, a diagram with several pages can be saved in several files, with disjoint sets of pages, thus enabling several users to edit them independently. The Join command simply merges two SDL diagrams of the same kind.

Using CM Synergy Together with an SDL System

This section describes one way of using the SDL suite and CM Synergy in an integrated way.

For more information about the SDL suite/CM Synergy Integration see the `Readme` file in the installation
`<SDL Suite inst dir>\examples\cm\win32\cmsynergy\` or
`<SDL Suite inst dir>/examples/cm/unix/cmsynergy/`.

For a more general description of using a configuration or version control system, please see "How to Manage the Diagrams in a Project" on page 155.

Introducing CM Synergy with the SDL suite - Migration

When handling of files by CM Synergy is to be applied on a system being developed by the SDL suite, the following approach can be used.

1. Make sure that all files (related to the system) are configured in one directory hierarchy outside CM Synergy.

2. Install the CM Synergy menu into the Organizer. Add `cmsynergy.ini` to your `org-menus.ini` file.

   The SDL suite will search for the `org-menus.ini` first in the directory where the SDL suite was started, then in a directory pointed to by the HOME environment variable and finally in the directory in which the SDL suite was installed.

   If you do not already have a `org-menus.ini` file, the `cmsynergy.ini` can serve as one. Just copy `cmsynergy.ini` to either your HOME directory or to where you have the SDL suite installed and rename it to `org-menus.ini`. For more information on dynamic menus, see "Defining Menus in the SDL Suite" on page 18 in chapter 1, *User Interface and Basic Operations*. (The CM Synergy menu that comes with the distribution of the SDL suite is an example and could be tailored by the user.)

3. Set your path environment variable to include

   – `<CM Synergy inst dir>\bin` **or** `<CM Synergy inst dir>/bin`, and
   – `<SDL Suite inst dir>\bin\wini386` **or** `<SDL Suite inst dir>/bin`.

   This is necessary to be able to start the tools from each other directly.

4. Start the CM Synergy Client.

5. Change role to ccm_admin.

6. Display the *Admin>Type Definition* dialog to define an SDL file type. Enter the following values:

   Type Name: SDL
   Description: Binary file
   Super Type: binary
   Initial Status: working
   Require Task at:<none>

   Verify Comment Existence on Promote / Check In: Off
   Allow Update during Model Install: Off
   Associate with a File in the File System: On
   Can be a Product File: Off

   Icon Color / Fil: binary.bmp
   File Name Extension: .ssy

7. Display the *Type>Modify File Operations* dialog, and enter the following values:

   (If necessary, replace `sdt` below with the command you use to start SDL Suite. If the sdt script is not in your PATH, you should specify a full path.)

   Type Name: SDL
   Description: Binary file
   Command Templates:
   Graphical User Interface:

   Edit: sdt %file1

View: sdt %file1
Compare: sdt -fg -grdiff %file1 %file2
Merge: sdt -fg -grdiff %file1 %file2 -mergeto %outfile

Command line interface:
Edit: sdt %file1
View: sdt %file1
Compare: sdt -fg -grdiff %file1 %file2
Merge: sdt -fg -grdiff %file1 %file2 -mergeto %outfile

Print:
Compare Attribute: source

8. Click *OK*.
9. Click *Update Type*.
10. Define a project file type (*.sdt) by entering the following values:

    Type Name: SDT
    Description: SDT project file
    Super Type: ascii
    Initial Status: working
    Require Task at:<none>

    Verify Comment Existence on Promote / Check In: Off
    Allow Update during Model Install: Off
    Associate with a File in the File System: On
    Can be a Product File: Off

    Icon Color / Fil: ascii.bmp
    File Name Extension: .sdt

11. Display the *Type>Modify File Operations* dialog, and enter the following values:

    (If necessary, replace `sdt` below with the command you use to start SDL Suite. If the sdt script is not in your PATH, you should specify a full path.)

    Type Name: SDT
    Description: SDT project file

Command Templates:
Graphical User Interface:

Edit: sdt %file1
View: sdt %file1
Compare: %FAIL
Merge: %FAIL

Command line interface:
Edit: sdt %file1
View: sdt %file1
Compare: %FAIL
Merge: %FAIL

Print:
Compare Attribute: source

12. Click *OK*.
13. Click *Update Type*.
14. Close the dialog. (*File>Close*).

    Now you have set up CM Synergy for SDL system diagram and SDT project files.

15. To manage the other SDL diagram types select *Tools>Migrate>Options>Set>Edit* to open up the migrate rules file (`migrate.rul`) in a text editor.
16. After the entry for your new type which should look something like (might differ depending on platform):

```
MAP_FILE_TO_TYPE .* [Ss][Ss][Yy]$ SDL # Created auto-
matically ...
MAP_FILE_TO_TYPE .* [Ss][Dd][Tt]$ SDT # Created auto-
matically ...
```

    add the following lines:

```
MAP_FILE_TO_TYPE .* [Ss][Bb][Kk]$ SDL
MAP_FILE_TO_TYPE .* [Ss][Pp][Rr]$ SDL
MAP_FILE_TO_TYPE .* [Ss][Pp][Dd]$ SDL
```

```
MAP_FILE_TO_TYPE .* [Ss][Uu][Nn]$ SDL
MAP_FILE_TO_TYPE .* [Ss][Oo][Pp]$ SDL
MAP_FILE_TO_TYPE .* [Ss][Ss][Tt]$ SDL
MAP_FILE_TO_TYPE .* [Ss][Bb][Tt]$ SDL
MAP_FILE_TO_TYPE .* [Ss][Ss][Uu]$ SDL
MAP_FILE_TO_TYPE .* [Ss][Pp][Tt]$ SDL
MAP_FILE_TO_TYPE .* [Ss][Vv][Tt]$ SDL
MAP_FILE_TO_TYPE .* [Ss][Ss][Vv]$ SDL
MAP_FILE_TO_TYPE .* [Mm][Ss][Cc]$ SDL
```

and save the file. Be very careful with the syntax in this file as it is sensitive to syntax errors. Close the open dialogs.

17. Use the *Migrate* facility to load the files into CM Synergy, via *Tools>Migrate*.

    – Put in the path to your Project directory in the *From Directory* field, or use the browse button to locate them.

    – In the *Project* field use the syntax ProjectName-Version, make sure to set the version to a meaningful value.

    – Next click on the menu item *Options>Set>Set Object State To>released* and click *OK*.

    – **Windows only:** To be able to change files you must set the option *Make Copies Modifiable* in *Work Area Properties* and then click *OK*.

    – To ensure that the correct files are being migrated it is suggested that you click *Preview*. If the results are what you expected go ahead and click *Load*.

    Now you have a project that is baselined in a released state which you can start to work from.

18. Each developer will now need to set up their own working environment before using CM Synergy with Telelogic Tau.

Introducing CM Synergy with the SDL suite - Set up your working environment

After the build manager has migrated your SDL system into CM Synergy, each developer will need to set up their own CM Synergy work-area. This should be done using the CM Synergy client to ensure that all the correct options are selected.

1. Select the correct project baseline for your SDL system as directed by your build manger.
2. Check out your own personal working version of the project hierarchy using the correct options as directed by your build manager, check out the system file (this will be your own personal version).

> **Note:**
>
> You will normally only have to set up a working environment in this way once for each major release of your software.

3. Load the CM Synergy menu into the Organizer <org-menus.ini>

Introducing CM Synergy with the SDL suite - Day-to-Day Working with CM Synergy

These steps assume that you will be using the (recommended) CM Synergy Task Based-CM methodology. They may be executed from the Telelogic Tau Organizer.

1. Select your SDL system and start CM Synergy from the Organizer.
2. Set your default task (also known as the current task), create one if required.
3. Check out the file(s) you wish to work on.
4. Edit and test as required.
5. Check in your (default) task.

> **Note:**
>
> If you are working as part of a team you may pick up your colleagues' latest work by selecting the top level directory and using the *Update* command (from time to time you may need to do a full update from the CM Synergy client - ask your build manager).

Using ClearCase Together with an SDL System

This section describes one way of using the SDL suite and ClearCase in an integrated way. For more information about SDL suite/ClearCase Integration, see `Readme` files in
`<inst dir>\examples\cm\win32\clearcase\` or
`<inst dir>/examples/cm/unix/clearcase/`.

For a more general description of using a configuration or version control system, please see "How to Manage the Diagrams in a Project" on page 155.

Introducing ClearCase with the SDL suite – Checking in Files

When handling of files by ClearCase is to be applied on a system being developed by the SDL suite, the following approach can be used.

1. Make sure that all files (related to the system) are configured in one directory hierarchy outside ClearCase. See the work area part in Example 50 on page 156.

   – Note that when working with ClearCase, the original area and work area point to the **same directory** – the top level directory for the diagram system.

2. Install the ClearCase menu into the Organizer. Please add clearcase.ini to your org-menus.ini file. The SDL suite will search for the org-menus.ini first in the directory where the SDL suite was started, then in a directory pointed to by the HOME environment variable and finally in the directory in which the SDL suite was installed.If you do not already have a org-menus.ini file, the clearcase.ini can serve as one. Just copy clearcase.ini to either your HOME directory or to where you have the SDL suite installed and rename it to `org-menus.ini`. For more information on dynamic menus, please see "Defining Menus in the SDL Suite" on page 18 in chapter 1, *User Interface and Basic Operations*. (The ClearCase menu that comes with the distribution of the SDL suite is an example and could be tailored by the user.)

3. Set an appropriate ClearCase view. Copy the system file to the top level directory of the ClearCase file system. You may have to edit it in order to remove the line defining `SourceDirectory`.
4. *Open* the system file to bring up the structural view of the system (the diagrams are marked as invalid in the Organizer – this is OK for now).
5. The *MkDir for Object* menu command can be used to create the directory structure in a ClearCase VOB. Select an object in the Organizer and execute the *MkDir for Object* menu command to create the directory for the selected object in the ClearCase VOB.
6. Now you can populate the ClearCase directory structure with the diagram files, by copying the files from the directory in step 1. above.
7. Re-open the system file in the Organizer. All diagrams should now be connected to their files.
8. Select the *System File* icon and do the *Recursive MkElem* menu choice to create all the objects in the ClearCase VOB.
   - Note that no ClearCase element is created for the system file.
9. Select the *System File* icon and the *Recursive Check In* will check in all objects into the ClearCase VOB.
10. The directories must be checked in separately. Use the command *Check In Directory* to do that.

The system file for the diagram system can be checked in but it is not suitable for version control since the Organizer wants to update it in situations unrelated to revision changes. The system file should be regarded as one developer's personal view of the system being developed. On the other hand, the top level control unit file should be checked in and is suitable to be put under revision control.

Introducing ClearCase with the SDL suite – Opening a System

The top level control unit file allows to load the system into the Organizer if a user starts from scratch. Say that there is a checked in diagram

system in a ClearCase VOB. A developer that wants to start working on that diagram system has to do the following.

1. Mount the ClearCase VOB containing the diagram system.
2. Set the appropriate ClearCase view and start the SDL suite in it on a new system.
3. Load the ClearCase menu into the Organizer
   `$telelogic/sdt/examples/ClearCase/sdtcc.mnu`
4. Set the *Source Directory* to the top level directory for the diagram system.
5. *Connect* the top level control unit file with the *System File* icon and run the *Recursive Update ClearCase* menu command. This loads the system into the Organizer.
6. *Save* the system file. (The Organizer will warn that the top level control unit file is read only but this can be disregarded.)

Chapter 4 Organizing a Project