[ICUKL November 2002, Goa, India]

Utilization of the Visual Studio IDE for Integrating the UNL Development Environment

Gunarso
Agency for the Assessment and Application of Technology (BPPT)
Indonesia
gunarso@aia.bppt.go.id

Abstract

UNL Center provides UNL developers with a set of tools to develop a specific language module called UNL Development Set. This set contains the DeConverter and EnConverter software, Word Dictionary Builder, Co-occurrence Relation Dictionary Builder, etc. Each tool needs some inputs and gives a certain output, which are needed to be integrated and maintained continuously during the development phase. Since the UNL Center do not provide the integrated development environment, the utilization of the Visual Studio IDE, that provides a way to organize the tools and related files, enabling an efficient, simple, and faster language module development.

1. Background

The UNL system consists of UNL Language Servers, UNL Editors and UNL Viewers. UNL language servers consist of a specific module of specific language to make the conversion task between this language into UNL representation and vice-versa.

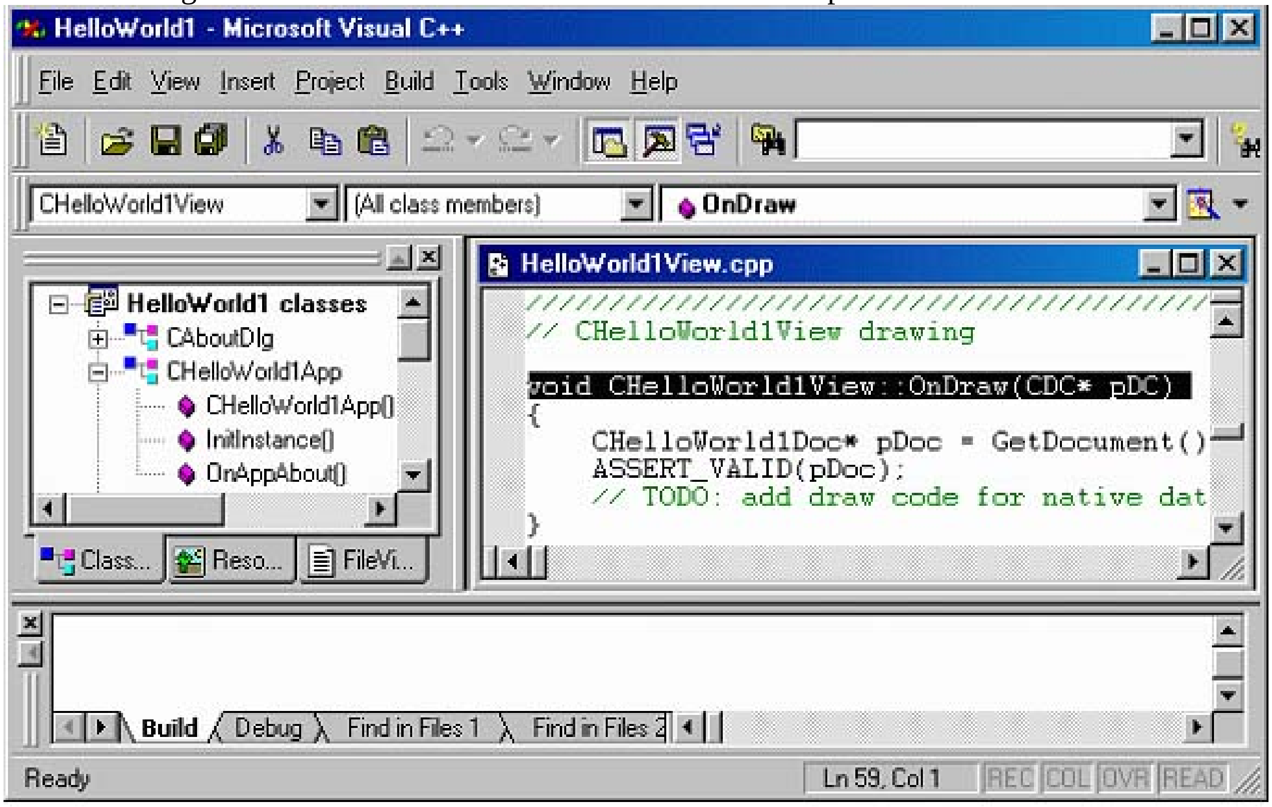
To create a new UNL language Server of specific language, UNL Center provides DeConverter and EnConverter software, which is independent language, as well as another facilitating tools. All of this software, called UNL Development Set, available to the UNL Developers who have signed the UNL Development Set agreement with UNDL Foundation. Although developers can get the necessary tools for creating a new language server, but the user interface and development environment are insufficient and unsatisfying. For example, during the DeConverter rule tuning, it was involving many input files like word dictionary, UNL sentences, grammar rule, and output file. The needs to open all those files and modified it as necessary are high, but the integrated development environment is unsupported by the UNL Center. Currently, developers are use the existing editor in their PC to do the editing job, and running the DeConverter or EnConverter as needed on the other DOS prompt window.

For faster grammar rule development, it was needed to have an integrated development system, which consists of file editor, application activator, and a way to manage the overall files. This system is responsible for the easier user interface and file management.

Regarding this problems, we try to utilize the Microsoft Visual Studio IDE (Integrated Development Environment), that provides a way to organize the files, tools, and highly customizable, as an integrated development environment for UNL System. With just developing one simple tool for automatically executing the DeConverter or EnConverter that we called *autounl*, the grammar rule development environment can be integrated and run easily.

2. Visual Studio IDE

Visual Studio IDE is an integrated development environment that is designed to make a programmer’s life easier. It includes tools and features for managing a project's source code and configuration setting and for visually designing its resources, an editor programming language syntax, and even an integrated debugger that can step through code executing in a browser.



```
/////////////////////////////////////////////
// CHelloWorld1View drawing

void CHelloWorld1View::OnDraw(CDC* pDC)
{
    CHelloWorld1Doc* pDoc = GetDocument()
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native dat
}
```

Fig 1. General Appearance of Visual Studio IDE

The Visual Studio is a fully integrated development environment (IDE). From within the Visual Studio IDE programmers can do anything they could possibly want to while developing a program. In the earlier of programming, a programmer used to rely on a collection of different tools. Each programmer had a separate editor, a compiler, a debugger, and other tools as necessary. With an IDE, not only are these tools collected into one environment but they also are designed to work together in order to improve the productivity of the programmer.

The Visual Studio Integrated Development Environment (IDE) is organized into four areas: the menu bar and a set of toolbars, the project view window, the source code editor, and the debug window.

The project view window gives programmer an overview of all items in his project. Programmer can look at his project through three different views: the class view, the resource view, and the class view. The file view gives a list of all source files, header files, and resource files in the project. The class view provides a list of all classes in the project with all functions in those classes. A note has to be made that any change made in file view will not be shown or available in the class view until the project is rebuilt or saved.

The debug window displays information during the project building process. Here programmer will see error and warning messages, if there are any. The debug window is also used when executing a project in debug mode.

During the development, programmer work on a single application as a *project*. A project is a collection of files: source, headers, resources, settings, and configuration information. Visual Studio is designed to enable work on all aspects of a single project at once. Programmer creates a new application by creating a new project. When he wants to work on his application, he opens the project rather than open each code file independently.

The other advantage of the Visual Studio IDE is highly customizable. Programmer could easily add a new tool in the environment, and then assign a button through the toolbar customization, and put it in the menu bar. Then programmer can run the tool by simply clicking the related tool button.

In relation with UNL development environment, we try to utilize the Visual Studio IDE in creating workspaces as a one single UNL development project that is contains the DeConverter or EnConverter input and output files. UNL developer can edit all input files such as grammar rules, UNL representation, word dictionary, and co-occurrence dictionary using the source code editor. Likewise, the output file can be browse using the source code editor. As for the Deconverter or EnConverter itself, we customize it as a tool and then assign a button and put it in the menu bar. The UNL developer can run the Deconverter or EnConverter by simply clicking the associated button. By utilizing this IDE, the developer can improve the grammar rule productivity.

3. Embedding UNL tools in the Visual Studio IDE

UNL Development Set tools consist of DeConverter and EnConverter software, word Dictionary Builder, Co-Occurrence relation dictionary builder, and other tools. These tools are being executed from the DOS prompt command by entering the program name and its input and output arguments. The input and output files are edited separately using the existing editor.

The philosophy of UNL Integrated Development Environment is embedding the UNL Development set into the Visual Studio IDE toolbar menu, and utilizes the source code editor to edit the input files, so we can take the advantage of the Visual Studio IDE. Thus, firstly we must automate the execution process before we attach it into the environment.

DeConverter and EnConverter normally executed through the DOS prompt window. When it was executed, a window will appear and user must click the run button that exists in the DeConverter or EnConverter window. This process is annoying and wasting the developer time in developing the grammar rule. To safe the development time, we create a special tool in purpose to click the button automatically during the execution of DeConverter or EnConverter software. This tool, which we called *autounl*, will execute the DeConverter or EnConverter along with its input and output files according to the autounl configuration file named **autounl.cfg**. Autounl will read the necessary input files from the configuration file and pass it to the DeConverter and automatically click the **runDeco** button, and finally pass the result into the related output file.

The typical autounl configuration file consists of the application name that will be executed, root directory of this application, inputs and output files, and button series that will be clicked automatically by autounl. Following is the example of the autounl.cfg file:

```
#--- SAMPLE CONFIG ----------------------
Root Directory: C:\UNL_DATA\
Application: DeCoL27.exe iuw.dic idgrul.txt ?[4|unlnews3.txt] out_tdh.trc
-c crid.dic -l ?[3|4] -s ?[1|1] -n ?[2|1]

Run As: 4
Max Wait: 120
Dialog Title: DECOL27
Application Title: DeConverter
#--------- Key -------------------------------
Click Button Name: Setting
Wait Window: ,,Setting
Wait: 100
Click Button ID On Dialog: 03F9, Setting
Wait Window: ,,Open
Click Button Name On Dialog: &Open, Open
Click Button Name On Dialog: OK, Setting
Wait: 100
Click Button Name: DeConvert
```

The above configuration means, when the autounl is run, it will call the **DeCol27.exe** that will take the iuw.dic, idgrul.txt, and crid.dic as input files, and assign filename out_tdh.trc as output file. Furthermore, autounl will click the buttons that are specified in the configuration file.

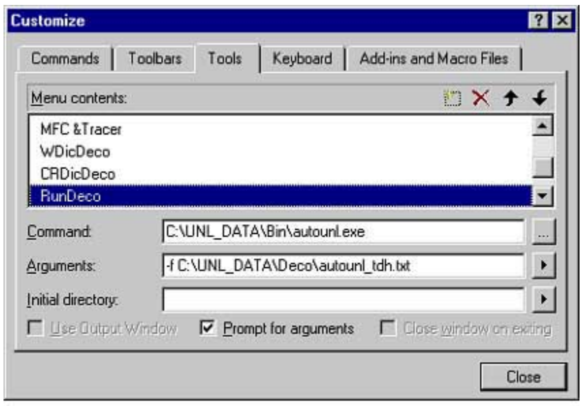
According to the behavior of rule developer when adjusting the rule, that they will process a sentence one by one, or a group of sentence together in the highest trace level, and then process all corpora at one in zero trace level. We facilitate the user to input those variables as autounl arguments. There are four arguments for autounl and each argument has a default values. Those arguments consecutively are as follows:

1. Starting Sentence to be deconverted. Default value is 1.
2. Number of sentences to be deconverted. Default value is 1
3. Tracing level. Default value is 4
4. UNL document file that will be processed.

Using this scheme, autounl can be run without argument or up to four arguments as needed. If the developer want to process and trace a certain sentence in the default UNL document, then he should input the sentence number as argument. If he wants to process sentence number 7 until 9 with default trace level (4), he only input two arguments 7 and 3 (three sentences starting from sentence number 7).

Now, we are ready to add the autounl into Visual IDE environment as a new tool. This can be done by clicking the **Tools** menu and then choose **Customize** sub menu. The customize window will appear as shown in the figure 2.

Choose **Tools** menu in the Customize window, and then type the application name in the menu contents. Put the autounl.exe executable file in the **Command** form and autounl configuration file in the **Arguments** form as shown in the figure 2. Tick **Prompt for arguments**, so the user can input the needed arguments for the autounl.exe.

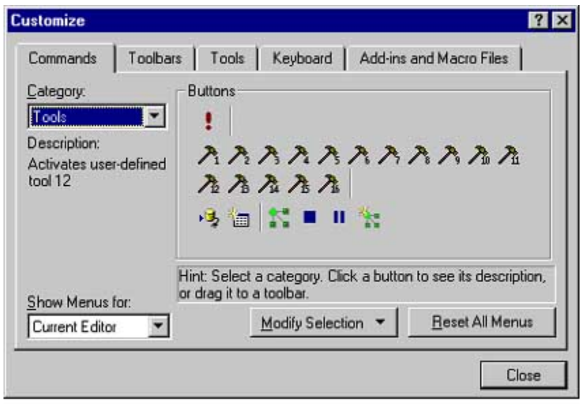


Command: C:\UNL_DATA\Bin\autounl.exe
Arguments: -f C:\UNL_DATA\Deco\autounl_tdh.txt
Initial directory:
[✓ Prompt for arguments]

Fig 2. Embedding *autounl* into IDE

At this point, we should bring this tool into the main visual IDE menu. This can be done by choosing the **Commands** menu in the Customize window. Select **Tools** from **Category** menu. This selection will bring out tool buttons (hammer icon and a number that relating to the existing tools) in the view as shown in the figure 3. Click and drag the related autounl tool into the Visual IDE main menu. We can change the button appearance by clicking the right mouse over the tool icon in the main menu and selecting **Button Appearance**. Buttons selection will appear and we can choose the desired button as the DeConverter button. Now we can run the DeConverter or by simply clicking the associated button in the menu bar.

Likewise, we can embed the EnConverter and the word dictionary builder into the Visual IDE environment, and place it in the menu bar. So, all UNL tools have been embedded into the IDE tool bar as it was shown in the figure 4.



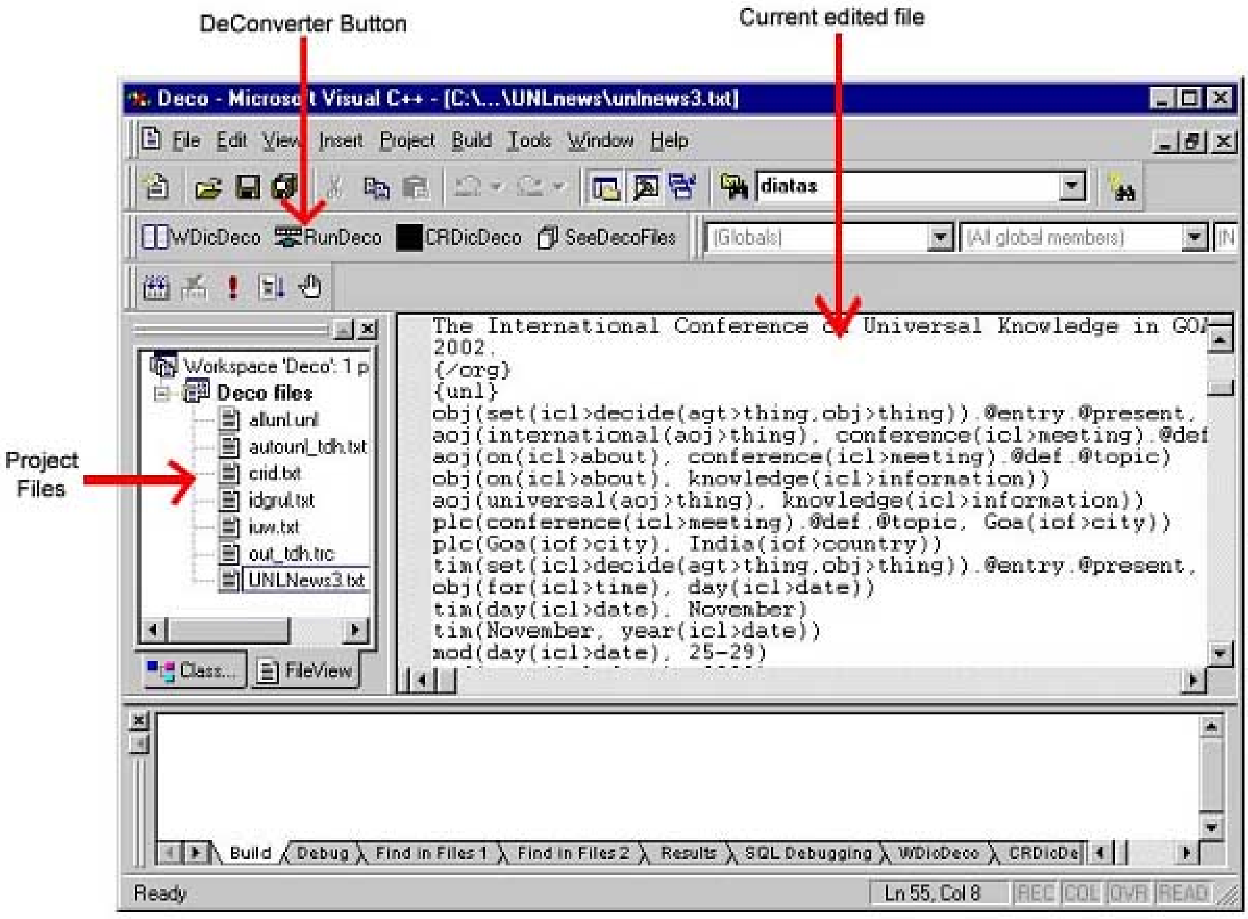
Category: Tools
Description: Activates user-defined tool 12
Hint: Select a category. Click a button to see its description, or drag it to a toolbar.

Fig. 3. Create tool button in the Visual IDE main menu.

4. Running UNL tools and organizing files in the IDE

Once we have embedded the UNL applications into the environment, we can run it by clicking the associated button. However, we still need a way to manage a set of input and output files in the UNL development environment. To do this, we can utilize the workspace and project scheme as if we are creating a programming project in C++ or J++.

To create DeConverter workspace, first run the Microsoft Visual C++ or J++ that we have customized for UNL development, then select **New** from **file** menu. This will display the New window. Select workspace menu from New window and specify the workspace name and location. The new workspace will appear in the workspace area. To insert new rule development project, click the new workspace using the right mouse button and select **Add New Project to Workspace ….** New window will appear again, and select **Utility Project** for the type of the project, then specify the name of the project. A project has created in the new workspace. Furthermore, we should insert all UNL development files like rule, word dictionary, UNL sentences, and output files, into the project. From this point, we have finish in creating a UNL development project and ready to run DeConverter and adjust the DeConverter rule. This setting can be used again if we want to work on the same project by opening the same workspace. We just open the workspace once, rather than open each UNL file independently.



```
The International Conference on Universal Knowledge in GOA
2002.
{/org}
{unl}
obj(set(icl>decide(agt>thing,obj>thing)).@entry.@present,
aoj(international(aoj>thing), conference(icl>meeting).@def
aoj(on(icl>about), conference(icl>meeting).@def.@topic)
obj(on(icl>about), knowledge(icl>information))
aoj(universal(aoj>thing), knowledge(icl>information))
plc(conference(icl>meeting).@def.@topic, Goa(iof>city))
plc(Goa(iof>city), India(iof>country))
tim(set(icl>decide(agt>thing,obj>thing)).@entry.@present,
obj(for(icl>time), day(icl>date))
tim(day(icl>date), November)
tim(November, year(icl>date))
mod(day(icl>date), 25-29)
```

Fig 4. UNL Development Environment

The view of the Visual Studio IDE after we creating a project is shown in the figure 4. In the project files area, we can see all UNL development files that we have inserted into the project. When we want to edit a certain file, we just click it and the content will appear in the editor window. We can edit it as if we write a program in C++. One area, the debug window is unused. This window is actually used to display message during compiling the source code of a specific project in C++ or another programming language.

To run the DeConverter we can click RunDeco button. When the deconvertion process is finish, the result can be view in the editor window by clicking the associated output file in the project files window. We can analyze the result and determine if something incorrect. Furthermore, we can edit the grammar rule file by clicking it from the project view window, save it, and run the DeConverter to check the modification result. This process was done repeatedly until we got the best deconvertion result of the UNL sentences. All processes are run and organize in a single environment, the Visual Studio IDE.

To facilitate the rule developer with the graphical view of the UNL representation, we can embed the UNL Viewer Software that have been developed by Indonesian Language Center, into the IDE. The process is same with the autounl embedding process. This graphical viewer can help in analyzing UNL sentences during the rule debugging and rule developer can determine the existing errors faster

6. Conclusion

The Visual Studio IDE is highly customizable, and we can utilize it to integrate the UNL tools for the grammar rule development. This means that we do not necessary to develop a special integrated system for the grammar rule development.

The grammar rules developer can organize all files in a single environment and run the UNL tools easily. Using this IDE, the UNL rule development productivity can increase significantly.

7. References

Cohn, M., Rutten, J., and Jory, J., Using the Developer Studio, *Web Programming with Visual J++™*, Sams.net Publishing, Indianapolis, 1997

UNU/IAS/UNL Center, *DeConverter Specifications Version 2.5*, UNU/IAS, Tokyo, 2000.

Uchida Hirochi, Zhu Meiying, Tarcisio Della Senta, *The UNL, A Gift for a Millenium,* UNU/IAS, Tokyo, 1999.