# Multi-Platform, Script-Based User Interface

Plamen Paskalev, Vladimir Nikolov

***Abstract:*** *The problem of developing several user interface modules for an application is overviewed. The main problems and advantages of using the different technologies in this aspect are discussed. The paper describes an implementation of an extended user interface, based on scripts, its internal structure and organisation. The goals and the approach are discussed. An approach for intelligent, CLIPS-based control of the user interactions is suggested. An implementation of such a software system is presented.*

***Key words:*** *Computer Systems and Technologies, Intelligent User Interfaces, Object oriented programming, CLIPS, .NET*

## INTRODUCTION

One of the most important aspects of creating a new application is the design of the user interface. From customer's point of view, it should be functional, easy-to-understand and as simple as possible. From the developer's point of view, it must be robust, cover the latest tendencies in the software development and follow the logic of the application. It is very common situation in practice, when different versions for the same application have to be developed for different platforms or for different types of usage (Desktop application, Web, Pocket PC etc.). Even if the platform is the same (MS Windows for example), different types of applications require different approach and time-consuming adjustments of the user interface.
The introducing of .NET framework from Microsoft, besides the other advantages, simplifies development of Windows and Web-based applications, but there are still differences between user interfaces for both platforms.
Currently used development system for platform independent applications are based on the following principles:

- Busines logic and GUI interface separation;
- Extensive use of script languages for GUI description. The script languages used are platform independent;
- Extensive use of user defined templates in the description;
- Visual tools for design and script code generation.

An example of the development system, satisfying the described main princips of the applications design and development, is Microsoft Visual Studio .NET Framework.
The paper describes a different approach in building UI module of an application, based on a platform independent script file, which describes interface and interaction between the user and application.

## SCRIPT BASED USER INTERFACE

### 1. Multi-platform using .NET framework

One of the main goals of introducing the .NET was the creation of a common environment for easier development of Web and Desktop applications, services, etc. for different hardware platforms.
Using the .NET compiler, the source files are converted to Intermediate Language code, which will be executed from Common Language Runtime (CLR). The conversion to the machine code is performed from JIT (Just in Time) compiler on the target computer.

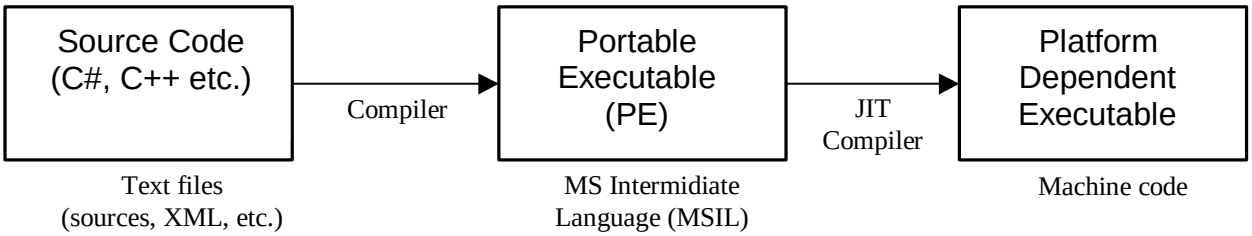


**Fig. 1**

Thus, one of main problems for developers of the user interfaces is solved: Using managed code, the application could be executed on different hardware platforms, using the corresponding version of the CLR.
There are some steps in direction of solving the other problem, too: The approaches for developing of Web and Windows applications are very close. But the developer still has to build different applications.

## 2. Script based UI

The idea, described in this paper is that the interface (as long as parts of the business logic of the application) could be defined in platform-independent script. User interface, which corresponds to the special characteristics of a platform, could be achieved, using this script.

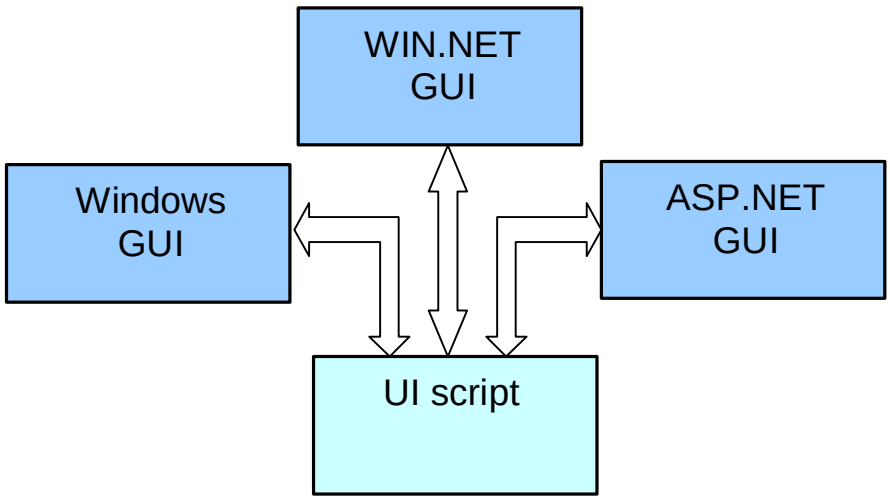


**Fig. 2**

There are some important steps to be solved to make this approach possible:

- The format of the script must be platform independent and, in the same time, flexible enough to contain all the needed information for different types of visualisation. The script can contain also description of some aspects of the user handling.
- The parsing of the script must be robust, fast and give the developers the opportunity to perform some steps for implementation of business logic, intelligent User Interface etc. The system, described in this paper is based on the well-known artificial intelligence tool - the expert system shell and language CLIPS as a development environment.

- There should be modules for visualisation of the interface for each target platform developed. As far as the CLIPS engine and the script itself will be used without changes in the different versions, the module, which is responsible for parsing, creation and handling of the user interface should be developed only. These modules are separate solutions, built and tested using the corresponding environments, which perform the platform-specific tasks and communicate with the CLIPS engine, which provides the connection to the business logic module, databases etc.

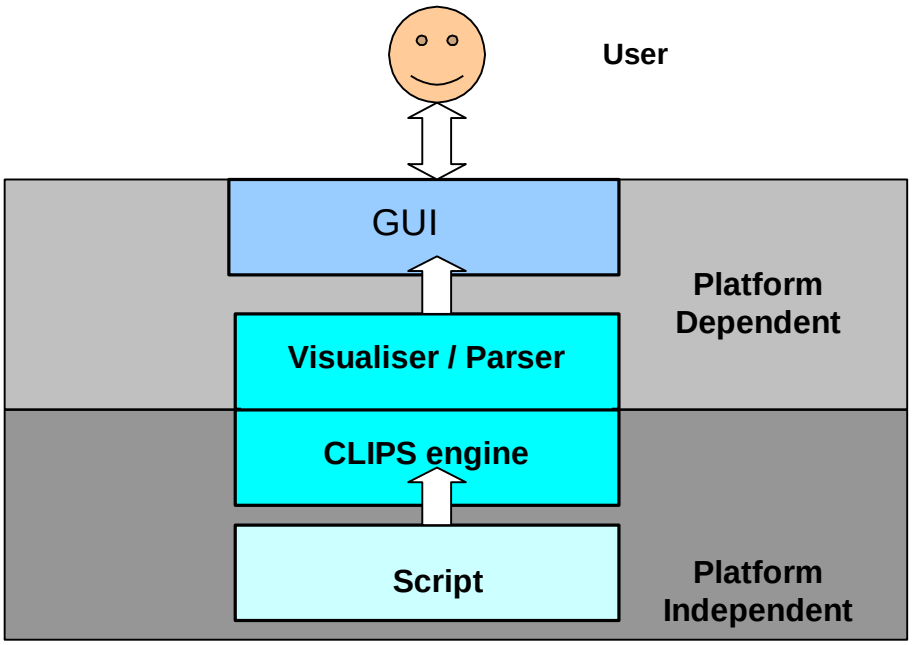


**Fig. 3**

The system, built in this way has the following advantages, compared to the standard and .NET approaches:

- Platform Independent – The user interface is built as a script, which is used without any changes in all the used platforms.
- Logic implemented: rules – In contrast to the procedural programming languages, this approach gives much more instruments for realisation of flexible user interface.
- Fuzzy calculation logic – CLIPS integrated fuzzy logic makes possible implementation of fuzzy rules in handling of UI.
- Artificial intelligence - intelligent control of the user interface. This point is discussed in more details later in the paper.

## 3. Intelligent UI Description

Using of the described system makes possible the implementation of some aspects of intelligent user interfaces in the application. The intelligent UI approach described satisfies main principles of application development, like the currently used distributed applications. A CLIPS script file describes intelligent user interface. The file contains static user interface parameters, for example:

- Control position on the screen;
- Control style and view parameters;
- Data bound sources.

The difference in our approach can be specified in the following main additional futures:

- The described approach is based on a nonprocedural programming language and AI. This language gives the full unlimited possibility for programming dynamic and self-organized interface.
- Programming Busines logic can also be writen in the CLIPS language. Using CLIPS rules, complex control interaction and data transfer between controls be programmed. This allows the programming of intelligent server controls for data binding, saving and changing information on the server.
- Template control parameters, initialy described in script are handled from script rules. This means that all control parameters can be changed dynamically, in real time, using knowledge base rules. For example, a rule can be specified to control the visibility of an interface element. In the existing systems this parameter can be changed in the run time, but using complicated algorithms. In the proposed approach more complex user activity can be analysed and added to the knowlege base of the CLIPS engine in the form of facts. In addition, the rule part of the knowledge base can be used in run time to change user interface, dinamically creating suitable interfaces for different users. Elements of a similar to the described interface are the intelligent sence elements implemented in the Visual Studio .NET development environment. By using them the user can browse all methods and properties of the selected control or class. This tool shows a list of possible values in a combo box control. Working with the tool is difficult, because the listed information is not classificated and structured. The group of programmers, working on the user interface are using one part of the listed data, and another group, working on the busines logic or on data access functionality – another part. To prevent this disadvantage, a rule-based approach can be used. The using of rules can formalise classiffications with the arbitrary logic. In addition, rules based on the user interaction (way of using the interface) can be created. The statistics of interface using can be analysed and thus only the necessary information could be displayed. For example, the combo box control contents can be sorted to show the most recently used information and, respectively, the unused information can be hidden.

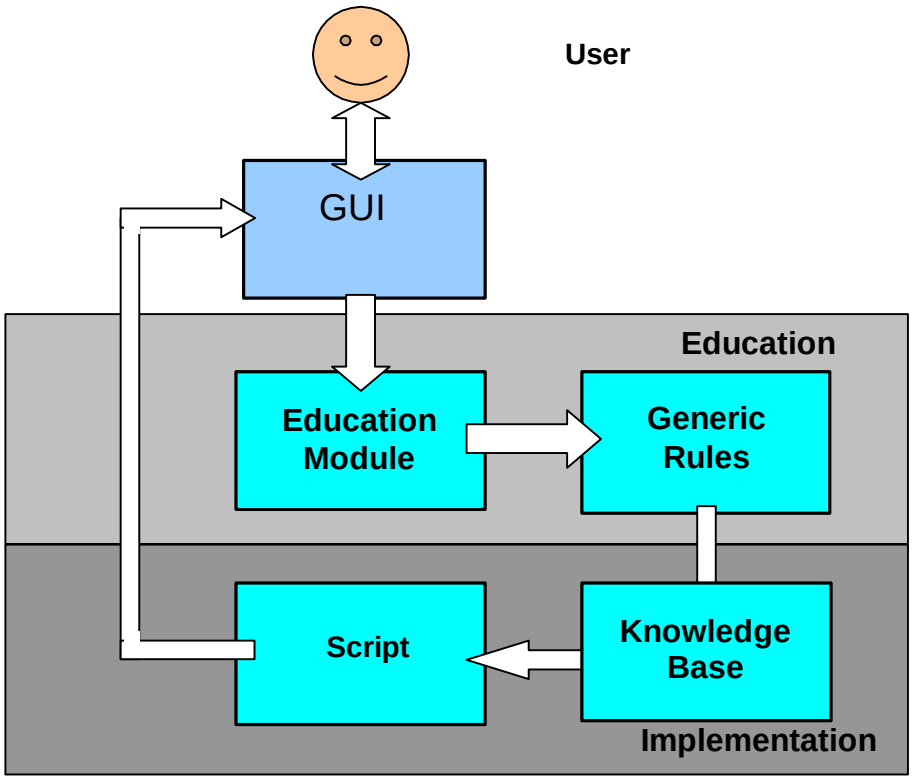


**Fig. 4**

Fig. 4 shows the learning mechanism implementation.
Two processing levels are separated – education and implementation. Both levels included CLIPS rules groups. High level rules analyse user interactions and calculate statistical parameters, used in low level – implementation.
A set of interface description templates is read by the Inference Engine and used to define the initial control set contents and settings. A corresponding set of rules is loaded into the rule base. Templates and rules are read from script files. Additional module, included Generic rules, collect facts about user behaviour, actions and decisions, and modify the existing scripts in real time.

## 4. System Implementation Description

Software modules are being developed, that implements the described approach. Parsers for following environments are currently available:

- Win32 applications;
- Windows .NET applications;
- ASP.NET.

The current program implementaton includes hierarchy of the following template data control groups:

- Root level template control. This is main control, included a set of dialog controls;
  - Dialog box template control, included a set of Group boxes;
    - Group box template control, included a set of following controls;
      - Integrated template Data controls – Data Grid, Chart graphic control;
      - Data select template controls - Listbox, Combobox;
      - Button controls - Push Button, CheckBox, RadioButton;
      - Data edit controls – Edit box, Spin, Date and Label;

Figure 5 shows implemented template control in action. The simple implemented applications are financial risk calculation modules, developed using different environments - Win32, Windows .NET and ASP.NET. All implemented application used CLIPS script format description.

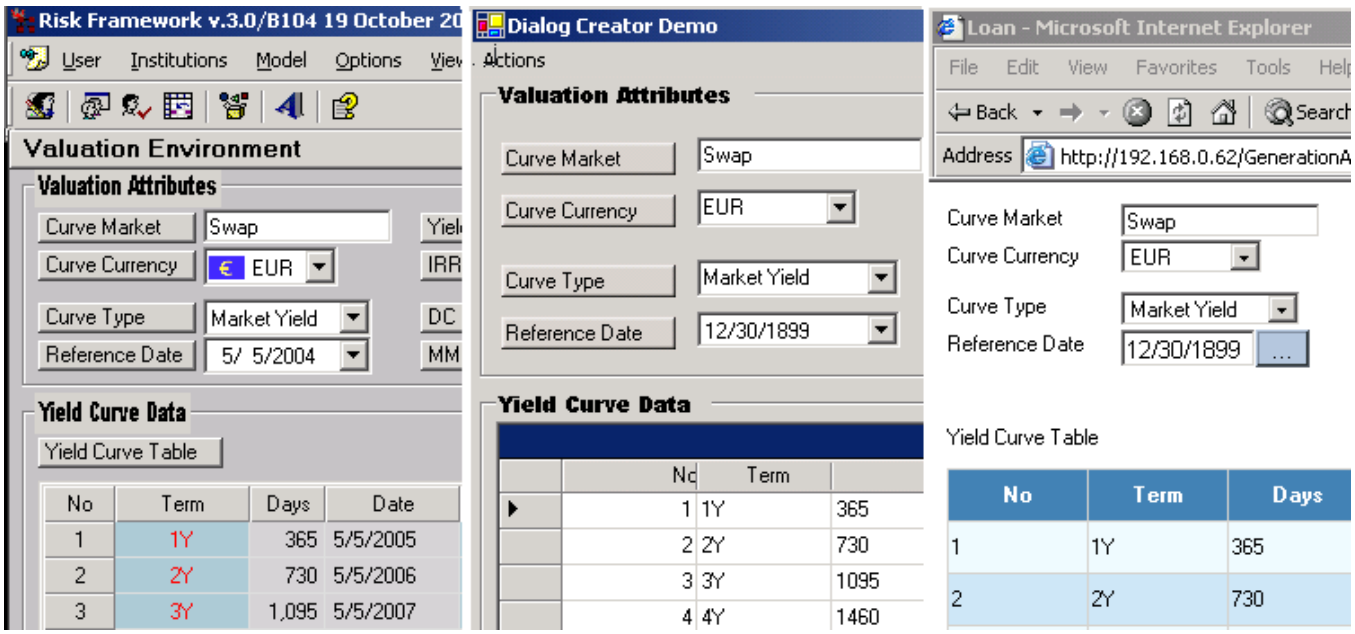


**Figure 5 a) Win32, b) Windows .NET, c) ASP.NET**

A parser for the Microsoft .NET Compact Framework (Pocket PC) environment is in process of construction. A visual tool for editing of User Interface scripts is in process of development.

## CONCLUSIONS AND FUTURE WORK

The practical application of the discussed approach has led to the following conclusions:

- The using of script based User Interfaces extends the flexibility of the software design process.
- Developing of platform - specific parsers gives the opportunity to use the same scripts for different platforms.
- The rule-based user interface module extends the handling of the user responses, adding some intelligent user interface features.

## REFERENCES

[1]. Esposito, Dino, BUILDING WEB SOLUTIONS WITH ASP.NET AND ADO.NET, Microsoft Press, 2002

[2]. Giarratano, Joseph C., Ph.D. CLIPS User's Guide Version 6.20, March 31st, 2002.

[3]. Kasabov, Nikola, K. FOUNDATIONS OF NEURAL NETWORKS, FUZZY SYSTEMS, AND KNOWLWDGE ENGINEERING, MIT Press, Cambridge, Massachusetts, London, England, 1996.

[4]. Microsoft MSDN

[5]. Richter, Jeffrey, Applied Microsoft .NET Framework Programming, Microsoft Press, 2002

## ABOUT THE AUTHORS

Dr. Vladimir Nikolov
Eurorisk Systems Ltd.
31, General Kiselov Str., 9002 Varna, Bulgaria
E-mail: nikolov at eurorisksystems dot com

Plamen Paskalev
Eurorisk Systems Ltd.
31, General Kiselov Str., 9002 Varna, Bulgaria
E-mail: ppaskalev at eurorisksystems dot com