COMPEL – Command Based Interpreter and Developer's Library

Elias N. Bachaalany

Aziz M. Barbar, PhD.

90% Complete

Problem Statement

1. GUI applications alone are not enough, by providing a scripting engine and command line interface to applications we will make them extensible and more powerful.
2. COMPEL was created to tackle:
   - The lack of flexibility of today's applications
   - To provide developers a library to facilitate the integration of scripting abilities within their applications

Specifications

1. Completely written in C++ language for speed and cross-platform portability
2. Seamless integration with most programming languages
3. Easy and simple language designed specifically for command line interfaces
4. Provides a rich set of commands through its built-in extensions
5. Extensive developer documentation and standalone scripting examples
6. Features an IDE (Editor + Debugger) for standalone scripting and computer language learning process
7. COMPEL can be integrated with most programming languages: .NET, C++, Delphi, VB6, etc.

Solutions

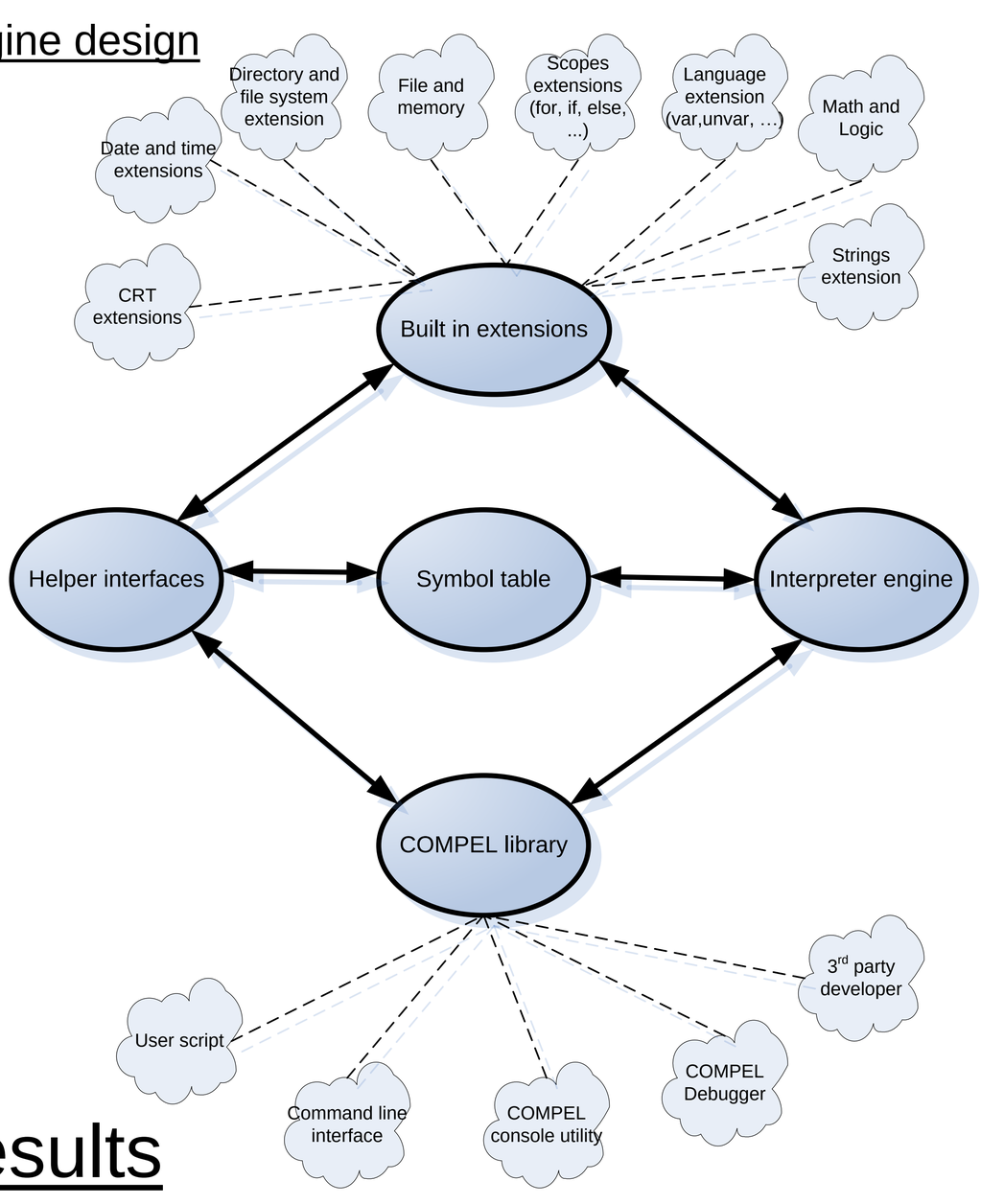
The concept of scripting engines and integrating them with applications is not something new, however the existing scripting languages impose a complex and non end-user friendly syntax.

COMPEL, the command based language is specifically designed in a user-friendly manner, so that even non-programmer can write a simple script without the constraints of strict syntax or hard programming language:

1. COMPEL is based on a simple grammar that is suited for command line interfacing
2. It allows you to empower GUI applications with a command line interface and scripting facility
3. It provides a rich set of developer functions (through COMPEL developer API) to allow control to every aspect of the scripting engine from your application
4. COMPEL comes with an decent integrated development environment (editor and debugger) allowing you to trace your scripts The debugger was fully developed using the COMPEL API interface without using the COMPEL engine source code
5. COMPEL is not a language by itself and does not have built-in commands or reserved words, thus it is extensible and allows high level of customization
6. It's built-in extensions along with its IDE can form a standalone language well suited for teaching programming concepts

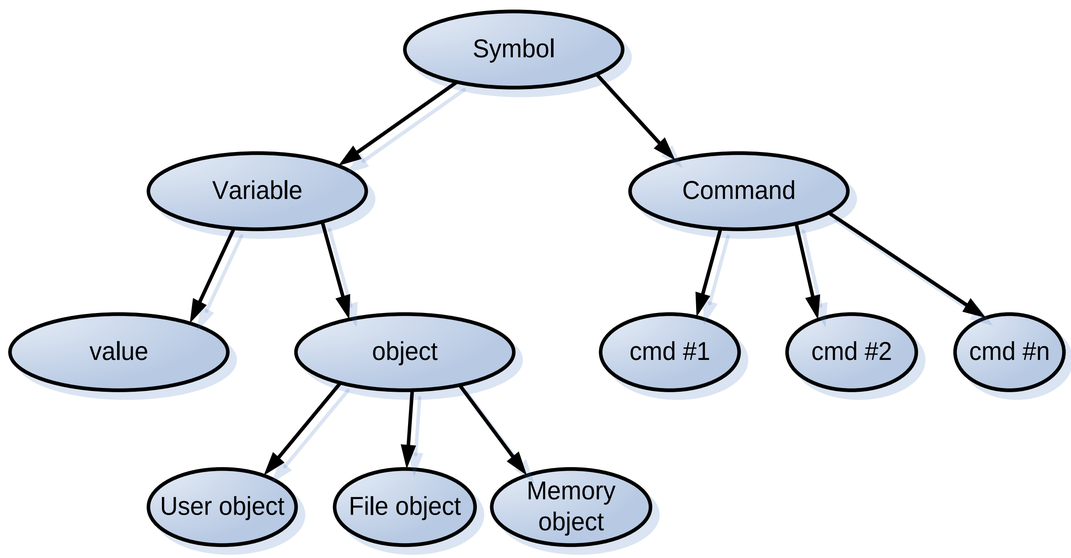
Design

Engine design



Symbol table design

Symbol table hierarchical view:



Symbol table tabular view (sample):

Symbol name	Symbol Type	Value
echo	command	(native code)
$pincode	value	316632
$record	user object	{age:18;name:elias}

Results

1. Applications can be extended through COMPEL
2. No need to code your own scripting engine (or parser), COMPEL will do the needed tasks for you
3. You can teach students programming concepts by designing and implementing your own command set
4. Your imagination is the limit when you use COMPEL

Conclusions

1. COMPEL's design and implementation made many things inherently possible. The symbol table hierarchical model allowed easy creation of language extensions for instance user commands, scopes, thread objects, etc.
2. The interpreter and parser can be improved by writing a faster pre-parser, by enabling embedded variable parsing and by the introduction of properties, functions and formulas into the engine.