

Concept – the technology, the language, the innovation

Eduard Șuică
RadGs Software,
Soveja 88, room 2,
Constanta, 900440, Romania
www.radgs.com
nospam@nospam.org

Abstract

The Concept technology for developing true RAD and web applications is described in this paper. Concept is a programming language with a framework for developing applications that run on a server and are instantiated on a client, reducing the work volume for both web and client/server applications. Three aspects are reviewed in this paper: the Concept programming language, the Concept Framework including the architectural model, and the performance superiority over other similar languages.

1. Introduction

The growth of the on-line application field seems to be unstoppable. Most of the applications developed these days are based on on-line technologies, from web services to on-line applications. With the advent of internet, it's almost a certainty that in the next 10 years all will be done on-line.

Concept is trying to offer the best solution to this problem with no compromises and full functionality. It manages to obtain good results, high speeds and low memory usage, these being the primary requirements of an on-line server.

The paper is structured as follows: section 2 describes the Concept programming language, section 3 discusses the Concept framework and model, section 4 the innovation brought by Concept. Several comparisons with other languages are presented in section 5. Conclusion and further work directions are outlined in section 6.

2. The language

The Concept programming language is the heart of the entire system. It is fully object oriented, with a traditional syntax, knowing that most of the programmers are reticent to new languages. Its syntax it's similar to Java/C#/C++/PHP, the programming language being developed to be "easy-to-use".

2.1. Data types

The data types used by concept are: number, string, array, object (class instance) and delegate.

Number and string are the basic data types that provide basic functionality. The number data type is represented as a 64-bit floating point number, considering that is enough for most cases.

The string data type is used to store strings, formatted and unformatted. It can hold up to 4 gigabytes of data, a string, or a data buffer. The formatting it's a little different that the usual kind of formatting, and borrowed from PHP. Let's say that we have an "Hello world" application that gets the name of an user, and then "hellos" him.

In an ordinary programming language we'll have something like that (in the best scenario):

```
str="Hello " + name;
```

Using Concept formatted string, we can say:

```
str="Hello $name";
```

We can even put whole expression in brackets like this "1+1 = \${1+1}", and there are a lot of gadgets available in the documentation of Concept.

These two (number and string) data types are considered static, and can easily be converted from one to another and compared without any type cast.

The array data type, is a little bit different. It's auto allocable, so the user doesn't need to provide the

“maximum number of elements”. It’s more like a very fast, block-allocable dynamic list, that can use string keys and a fast binary search for retrieving data. Each block can contain about 10,000 elements that can be accessed in O(1). This method reduces speed of array accessing comparing to “binary-type” arrays with about 5-8%, but provides a lot of functionality. Let’s see an example:

```
var[] some_array;
array[0]="some element";
array["test"]="someother element";
array["class"]=new SomeClass();
```

Concept arrays are untyped, so it’s possible to put anything in one, starting from numbers, to other arrays and objects.

The Concept object-type is the fundamental type used by the framework discussed in the next chapter. It stores instance of a class.

The object’s classes provides all the standards in use today:

- member access: private, protected, public
- abstract members: a function can have no body; to be defined later
- properties: a property is an alternative for calling SetProperty(x), GetProperty
- virtual function (events): a function can be declared as an alias for a function/member defined later (by a child)
- multiple class inheritance : each class can inherit one or more classes
- member overriding
- class constructor/finalizator
- operator overriding : you can define an operator (like in C++)
- static functions : a static function is a function that can be called without instantiating the class that owns it. This function contains no reference to “this” variable, where “this” variable is the same as in C++/Java/C# and represents the current instance of the owner object.

The Concept delegates are member functions stored into variables. The difference between other languages is that delegates are not required to be static members, and should respect the rights of the calling function.

Example:
var delegate=this.some_function;
delegate(parameters);

The Concept language is not an interpreter, neither a compiler. It combines these two, and it’s more like a virtual machine (like Java). This way can obtain the maximum speed with the maximum portability.

2.2. Memory management

The Concept memory management engine has two sub-systems: the SDL (Smart Data Linking) and the VLMS(Variable Linking Monitor Subsystem). SDL is similar to COM/.NET objects: each variable has a link count, and when it hits 0, the memory is freed. However is the problem of cyclic reference, that is when an instance contains a reference to another object that contains a least one reference to the initial object. For that, there is a mechanism similar to Java’s Garbage Collector, called VLMS. This way the memory management is both fast and clean.

In a few words, you don’t need to make delete after a variable was instantiated with new.

2.3. Exceptions

The exceptions used are multi-level, that means, that a function doesn’t catch an exception raised by the thrower, the function will forward it to the calling function, until a try/catch block will be encountered. If not, this will lead to program termination. As a difference from other languages, the catch block can’t be multiple due to the variant implicit type of variables in Concept.

2.4. Syntax differences

The syntax is traditional, based on Kernighan & Richie including most of the standard C keywords. The reason for using this grammar is very simple: it’s simple and efficient and the programmers like it.

However there is one operator not found in other languages, and is called “forced equality operator”, =&. This operator cannot be overridden, and it’s used as an assignment operator, when a class overrides the standard assignment operator (“=”).

Some other operators that appear in Concept and may not appear in other languages, are the typeof and classof operators, returning the type of a variable, and respectively the class of an object.

2.5. Interaction with other languages

Concept can interact with any other binary code, generated by any compiler that knows how to generate dynamic link libraries. Concept is using the C calling convention for libraries and it has libraries with most of the C functions, that can be used, wrappers for most of the common used libraries for accessing XML, ODBC and so on. As a note, each library must manage its own memory, and what is allocated, must be unallocated by the same library. For this, there are 2 library events, 2 functions, one called automatically when a library is loaded and the other one when the library is unloaded. All libraries are cross-platform, and can be used on most operating systems today.

3. The framework

The framework is a collection of over 100 classes for developing GUI based applications that run over network. The model is somehow different, and innovative.

3.1. Concept client/server model

The interface is instantiated on a client machine, while the code runs on a server, and they communicate between them using a TCP/IP connection. Let's use the model of the ASP applications: the code is running on a web server while the results are presented as a web page. Concept does the same, only it's not the web

client, it's the Concept client which is more interactive and is based on a continuous message exchange rather than the POST/GET methods of a web-based server (see figure 1).

For better understanding, we should think this way: If a database is an element of a database server then a Concept Application is an element of a Concept Server. In other words, Concept is an application server. This means that it's possible to develop client-server applications with all the code running on the server, using a generic client. This way the developing process on a client-server application is reduced with 50%: it's not necessary to develop the client. A client/server application has two components: the client, and the server; using concept you can develop client/server applications with only one component.

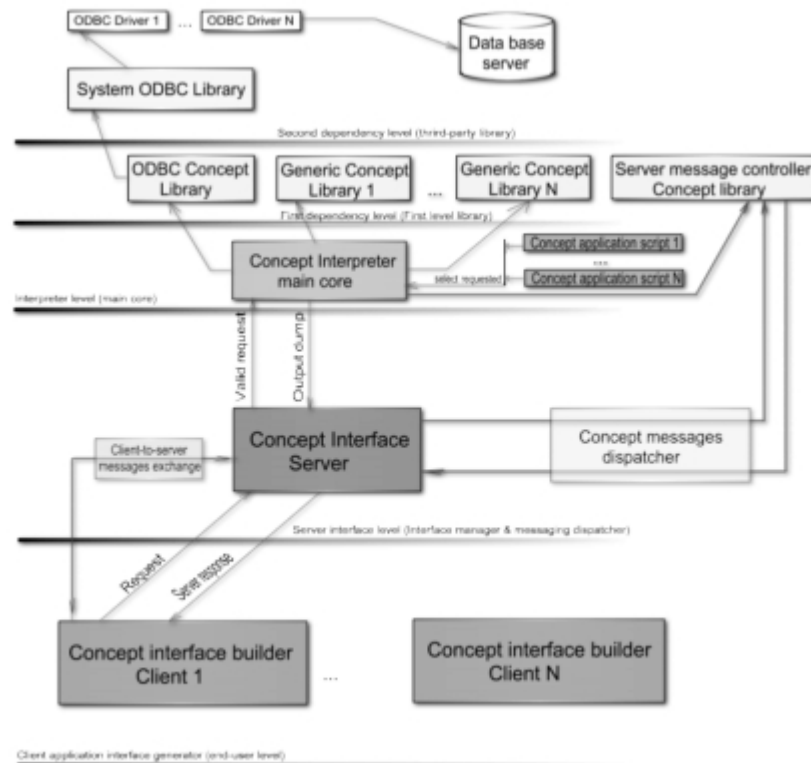


Figure 1. Concept system components and interaction between Clients and Server. By Concept System you should understand all the elements that are related to Concept (client, server, CGI, libraries, etc.)

However, Concept can be used to develop web-based applications using the CSP technology (Concept Server Page) similar to PHP and including all the standard technologies today like: AJAX, cryptographic algorithms, web sessions, database access and so on. For that, it has specialized classes for ergonomic usage and fast application development.

The Concept server is secured using RSA and AES cryptographic algorithms as a cryptographic system as so: the server has a pair of RSA keys (on 256-1024 bits, depending on the level of security selected), and at client connect it sends it's public certificate. The client generates a random AES key and sends it back to the server encrypted with server's public key.

Now both the client and the server have the same key, and the communication is now secured. In this way each session will have different keys. By using this method it is possible to minimize the work at the application security, this being managed by the Concept system.

The Concept client it instantiates applications “described” by the server. It’s build using GTKMM libraries, providing full portability.

GtkMM is a C++ wrapper for GTK libraries, used by the Linux family operating systems, and provides an excellent application base for developing cross-platform GUI applications. On the windows distribution, runs excellent, providing a nicer looking interfaces, that users tend to like.

Concept has over 60 controls, and the possibility of creating custom controls, as well as using ActiveX controls (on Windows only). The look of the applications is customizable using skins, to ensure user satisfaction.

3.2. Controls, and user classes

Some of the Concept controls are: RButton, RLabel, REdit, RTextView, RMenu, RPopupMenu, RToolBar and so on, including all the standard controls used by an application. The “R” preceding the control name its an acronym for “Remote”. Also Controls for database access via ODBC are included, like data sets and data objects. Concept controls are data-aware and they are very similar to Microsoft’s ADO objects. A lot of classes are available for FTP/HTTP access, XML, SOAP, even Google’s Search using their SOAP service, dialogs, printer services, file converters and so on.

3.3. Web development

Concept is used to develop web sites, in a manner similar to PHP, or ASP. Being faster than both PHP and ASP, it ensures a lower server usage, for the same amount of visitors.

Classes are available in the framework to create powerful web applications and managing sessions and content header based on the qDecoder library, that is available on most Linux systems and windows. The SACK (Simple Ajax toolkit) wrapper ensure a good interaction between concept and Ajax scripts. The cryptographic libraries provide access to algorithms used in user login (like md5, sha1, RSA) and in e-commerce, functions like hmac (used for on-line payments).

The SOAP library can help you develop SOAP servers and clients, at a high level, minimizing the complexity of the final product.

3.4. Online/client-server development

The structure of an on-line application is inspired by the Windows application technology. There is a message loop, that exchanges messages with the client, and with other applications. The client messages are received via the client socket on the server. Messages are divided into several categories such as properties messages, event messages and method invocation message. This messages are ensuring the data synchronization between the client and the server (managed in Concept, by the framework). When the users sets a handler for an event, the server informs the client that is interested on that particular event, and the client will send a notify message each type the mentioned event occurs. The purpose of this is to minimize the network traffic, and to allow Concept applications to use a very small bandwidth (less than 64kbps) for an acceptable interaction with the remote user.

The inter-application messages exchange is made via 2 sets of pipes, and a router provided by the Concept Server. Each applications has an APID (Application ID) that uniquely identifies the application. Using this system is possible to develop a chat program, for example, with a small number of lines, without using any socket or database in the design (the connectivity being provided by Concept).

The model of this type of applications is similar to telnet/ssh but offers full GUI interaction with a lower network usage.

The Concept client also allows custom controls that are executing on the client-server, but controlled by the application being executed on a server.

The developer environment used by Concept, called CIDE is written entirely in Concept, providing many features based on this technology. There is no need to use a CVS to keep track of the sources modified by multiple users and the entire team uses the same context.

An client-server application can easily interact with web applications. An Concept application can be informed by the events that appear on a web application, running on the same server, using the Concept Shared Memory library. This library is much simplified, and ported to a high level of interaction, and simpler management instead of the usual non-portable

shared memory operations that a programmer may use in a program developed on another system.

4. The innovation

Reinventing the wheel is impossible in today's context and Concept doesn't aim to be a revolutionary programming language. It implements all the today's features of a standard programming language, has a very traditional syntax, easy to write, making sure that a Java or a C# programmer doesn't have to spend any time learning the syntax. Theoretically any programmer with OOP experience should not have any problems adapting to Concept.

What is a little bit different, it's the compiler. Concept is cross-platform, it was developed on Windows using Linux technologies and tested on both Windows and FreeBSD 6.1, with very good results. For achieving this kind of portability, at the speed of a compiler, not of interpreter, the Concept compiler "pseudo-compiles" all the code, generating a 4-instructions code (called CAL=Concept Assembly Language) and one debugging instruction, using 4 virtual registers, OL, OR, AC, and ER meaning respectively operand-left, operand-right, accumulator register and extra (reserved) register. Before the code being translated into this form, it's performed an optimization, eliminating redundant code, and unused variables. After that, in the CAL form, some arithmetic expression are evaluated, partially translated into binary code, making from Concept an average speed programming language if we compare it to a compiler, and a very fast, maybe the fastest interpreter. Concept has practically a hybrid compiler. From this point of view, is similar to Java.

In the next section we'll discuss the benchmarks between Concept and other languages.

4.1. The innovative debugger

Concept has 2 debuggers: one that allows a Concept application to debug another one (very useful for the IDE) and another one that works with a debug server.

The first one has nothing special, and provides all the functionality of a basic debugger.

The debug server is providing a way to test an application run by an user on a remote

machine, in the same context as the user does. For this, the remote client informs the server that the application should run with a debug server and provides the address (the client must provide the address for security reasons), and the programmer, situated on another location, can see what caused an error or debug the application, using break points and watching variables as he normally does with a desktop application. For an "old-fashion" debugger, the programmer should generate the context where the bug reported by an user, and fix it. But that is not always possible, because many applications depend of the general hardware and software context of the machine. Using Concept remote debugger server, it's possible to debug an applications as it runs, being ran by the user himself.

5. Comparison with other languages

The test made, showed very good results for Concept. The tests included benchmarks and memory transfer tests. However it's hard to compare with other languages, because Concept is the first one of its kind (with code running on a server, and an interface on a client side). Concept was compared to PHP for relevance to Web Applications and to C++ for client/server applications.

Some of the benchmarks were made by the independent Euphoria developers for their interpreter (Euphoria is the fastest interpreter for now, but it's considered deprecated because of the missing class support, and limited extensibility). Concept was tested with Python, Perl and PHP. The rest of the interpreters benchmarks, are from the Euphoria web page.

The following results are made by Concept developers:

Interpreter	Iter.	String test	Call	Class inst.
PHP	6.330	11.266	7.341	20.167
Concept	1.302	9.944	2.302	8.782
C++	0.789	>200.000	~0.550	~0.850

The test were made on 12.000.000 iteration, 1.000.000 string concatenation, 1.000.000 function call with some generic arithmetic operations and 1.000.000 class instances, with constructor, on an under-clocked computer for better observation of the differences.

As you can see is a lot faster than PHP, but slower than C++, which is normal, due to full binary code execution of C++, the lack of advanced memory management. However, on

string operations, Concept is considerably faster than C++ and Java

Python was tested on win32, on iteration test, and the results were in favor of Concept significantly faster than Python.

Python is considered to be one of the fastest interpreters.

Using the report on the Euphoria interpreter home page (the iteration test is tested there) we can say that Concept is faster than python, perl, cyperl, ruby (more than 100% faster), elastic, and slower than Lua and Euphoria with about 50%. VBScript, JScript were not tested, because it's not relevant, due to their very slow execution relative to Concept.

Others comparisons were made with other languages, especially regarding the programming ergonomics, to port the Concept to the needs and style of the programmer, instead the other way.

The conclusion should be, that Concept is very fast, seen as an interpreter, and relatively slow, but comparable, seen as a compiler. The price of portability (to have code fully independent of the operating system it runs on) is a slow code relative to C++, but it manages to be faster than most interpreters that pretend to be "fast interpreters". At average, it runs on 1/5-1/6 compared to C++, which is enough even for time consuming tasks like Artificial Intelligence.

6. Conclusions and further work

As said before, Concept is not trying to reinvent the wheel, but to help developing of both client/server and web applications in a easy way, with a minimum effort and time. The lifecycle of an application is taken into account, and the applications maintenance should be very easy and friendly.

The Concept IDE is now in development, in beta stage, already providing many features, and ensuring that the programmer will not have a hard time when writing in Concept.

This technology will be distributed on a LGPL license or a special Concept license. For now, is free, and partial open source (excluding the kernel). All the Concept components developed in Concept are available for users to modify and improve. More libraries are ported each day, having, for now, about 1,000 functions in libraries. The idea behind Concept is to help the programmer as much as possible, to cover all the usual technologies, for the programmer to have a complete and structure framework at his disposition.

The future of Concept will be decided by the market. For now is in pre-release version. And the official distribution of Concept 1.0 is planned for later august 2006.

In this paper we have presented a novel technology for developing online applications.

Further efforts will be focused on:

- testing Concept in as many modes possible using real-life situation
- diversifying the framework
- developing a framework for web applications after the Ruby on Rails model

Acknowledgements

Concept can be downloaded from www.radgs.com (web site running on Concept).