The essence of Ruby

Wouter Borremans
0461911

November 18th, 2004

1 Preface

In this document i will give my personal view on the essence of the language Ruby which was first introduced in 1995 and commonly used in Unix environments. First i will have look at what Ruby exactly is and the history of the language itself. After that i will discuss the main features of Ruby and it's metamodel.

2 What is Ruby?

Ruby is the interpreted scripting language for quick and easy object-oriented programming. The language has many features to process text-files and to perform system management tasks (Like Perl). Ruby is available for free and makes you able to use it for free and distribute it as much if you want.

Ruby is a pure object oriented language, this means that everything in Ruby is an object. The language has a simple syntax which is based on Eiffel and Ada. Offcourse programming in Ruby needs programming skills, though users state that the understanding of the Ruby syntax can be easily learned. Ruby needs no variabele declaration (which you can compare with Basic or QBasic of the Microsoft Corporation) which makes you able to work quickly without worrying about variabele types etc. Ruby automatically typecasts the variabeles.

Another great feature of Ruby is it's great portabillity. At the moment Ruby is able to run on many types of UNIX, DOS, Windows 95/98/Me/NT/2000/XP, MacOS, BeOS, OS/2, etc. Ruby is most used on Linux OS based machines.
Why the name "Ruby"?
Influenced by Perl, Matz (The creator of Ruby) wanted to use a jewel name for his new language, so he named Ruby after a colleague's birthstone.

3 The history of Ruby

The language is created by Yukihiro Matsumoto in 1993. The basic idea of the language was to create a real object oriented based programming language (OOL) which filled the "holes" of Python and Perl which were not "real" OOL programming languages according to Yukihiro Matsumoto's believes.

In december 1995 the first version (0.95) was released on a Japanese domestic newsgroup. From that point on users became more and more enthausiastic using the language, this made it possible to start a community in 1996. Because of origin of the author of Ruby, all the documentation which was available was written in Japanese. As the community was growing, the translation of the documents to English was a fact in 1997. This made the popularity for this language even greater.

4 Ruby Features

This section will discuss some features for Ruby which i find most imporant looking at the essence of a specific programming language.

4.1 Commandline interpreted

Ruby is commandline interpreted, this means that every line will be executed line by line. A great advantage of this type of language is that you don't have to wait on the compiling of the program because you can immediately execute it. A disadvantage of commandline interpreted programs is that they execute at a much longer time then a precomilped program like e.g. Pascal or C++.

4.2 Object oriented

Everything in Ruby is an object. In practise this means that for example the number "1" is in Ruby an instance of the class "Fixnum". Ruby's object model was carefully designed to be both complete and open for any improvements. It has advanched features to add methods to a class -even during runtime- .

4.3 Distributeable

Ruby is able to communicate across a network with another Ruby program. In essence, this is not a feature which is built-in in the orgiginal package. This functionality is build by the Ruby community. Interconnecting Ruby programs opens a way to new projects where help of multiple systems is needed.

4.4 Untyped variabeles

As mentioned earlier in this report, the variabeles of a Ruby program are untyped. This means that the variabeles in Ruby act as placeholders, though the data is typed in the memory itself. Languages like C++ or Pascal check the type of the variabele at compile time. Ruby checks the type at runtime. No variabeles have to be declared, they're created at runtime.

4.5 Automatic memory management

One of the most important features of Ruby is automatic memory management (also known as garbage collection). In practise this means that you don't have to release allocated memory during runtime. No longer used memory where variables are pointing to are automatically cleaned up by the garbage collector. This has the following advantages:

- No memory leaks
- Less crashes and or errors which are related to memory assignment
- More efficient programming because you don't have to worry about memory assignment

The advantages mentioned below cause the program to run more slow at a rate of about 10%. In practise this amount is not something to worry about since Ruby a command line interpreted language is.

4.6 Advanced OO-concepts and features

Ruby supports several well known OO-concepts and features;

- *Singleton methods* are methods that are only given to a single object (e.g. A particular instance of a class must have a specific behaviour). These kind of methods are often used for elements of a GUI (Graphical User Interface) where different actions need to be taken when different buttons are pressed.
- *Operator overloading* makes the user able to give his own interpretation of an operator like "==" or "+=" etc.
- *Exception handling* Ruby offers two types of exception handling;
  - *Raise & Rescue* This method raises a error message which gives the user some more information on an exeption that happened. This method is used form the kernel class.
  - *Catch & Throw* This method makes you able to catch an error before the programs exits to give an error. After that you can continue running the program.

4.7 Datastructures

Ruby offers many types of datastructures such as dynamic arrays, hashes, strings, integers, bignum, complex etc.

4.8 Other (non technical) features of Ruby

One of the most important features of Ruby is that is is freely available. Many (online) libraries make it more easier to develop Ruby programs. Ruby is permanently developed and continuously keeps it's backwards compatibility.

Because of the fact that the language is very populair, many interfaces have been developed to for example Python, Java and Perl. Users have developed extensions to MySQL, PSQL MSSQL etc. This makes the language easily implementable in many organisations.

5 The metamodel of Ruby

This section describes the metamodel of Ruby. It shows the relations between the main (core) classes of Ruby.

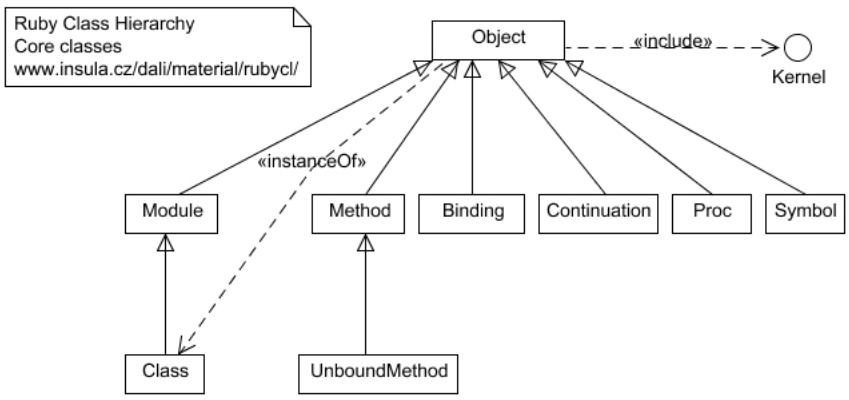

Figure 1: Relation between Ruby core classes

Each core method will be discussed below.

- *Object* is the parent class of all classes in Ruby. All the methods in this class are available to all objects. The object module mixes in the kernel module, making the built in kernel functionality accessible globally.

- *Module* is a collection of methods and constants. Methods in a module may be instance methods or module methods. Methods appear as methods in a class when the module is included, module methods do not.

- *Binding* objects encapsulate the execution context at some particular place in the code and retain this context for future use. The variables, methods, value of self, and possibly an iterator block that can be accessed in this context are all retained

- *Continuation* objects are generated by Kernel#callcc, they hold the return address and execution context, allowing a nonlocal return to the end of the callcc block from anywhere within a program

- *Proc* objects are blocks of code that have been bound to a set of local variables. Once bound, the code may be called in different contexts and still access those variables

- *Symbol* objects represent a Ruby name and is generated automatically using the :name literal syntax.

The structure of the metamodel of Ruby showed above is quite simple. All the classes mentioned are child classes of the parent "Object" class. All child classes inherit the functionallity of the parent. The object class has operating system functionallity due to the inclusion of the kernel.

6 Comparison between Ruby and Perl

In this chapter i will make a short comparison between Ruby and Perl to raise some advantages or disadvantages against eachother. To get a good overview of the differences between the language i created a table in witch all the important features are listed.

Ruby vs Perl	Ruby	Perl
Object orientation	Pure	Add-On / Hybrid
Static / Dynamic Typing	Dynamic	Dynamic
Generic Classes	N/A	N/A
Inheritance	Single class, multiple "mixins"	Multiple
Feature Renaming	Yes	No
Method Overloading	No	No
Operator Overloading	Yes	Yes
Higher Order Functions	Blocks	Yes
Lexical Closures	Yes (blocks)	Yes
Garbage Collection	Mark and Sweep	Reference counting
Uniform Access	Yes	No
Class Variables / Methods	Yes	No
Reflection	Yes	Yes
Access Control	public, protected, private	No
Design by Contract	Add-on	No
Multithreading	Yes	No
Regular Expressions	Built-in	Built-in
Pointer Arithmetic	No	No
Language Integration	C, C++, Java	C,C++
Built-In Security	Yes	Yes (perlsec)
Capers Jones Language Level	15	15
Distributed computing	Yes	No

The table clearly shows some disadavantages between Ruby and Perl. In my opinion Perl lacks to have support for several quite advanched methods like multithreading, class variabeles or methods which are more and morey wanted these days. This means that the designthoughts of these days meet the daily needs of programmers all around the world. Ruby even supports distributed computing, something perl never has heard of!

7 Conclusion

What makes Ruby a unique programming language? To form an opinion about that i first have to make sure what the word "essence" means for a programming language. In my opion, in this case "essence" means how the language distinguishes itself compared to other languages.

Ruby offers powerfull and easy to use syntax. The language is build on the "good" of several languages which make it very easliy implementable. Ruby distinguishes itself by being a onbject oriented language from head to tail. Due it's great portability it can be used in nearly all operating environments globally available. Ruby is able to is able to communicate over networks with other Ruby programs, this makes the language even more interesting for enterprices and smaller companies.

Wouter Borremans, November 2004

References

[1] Ruby, The object oriented scripting language, `http://www.ruby-lang.org/en/`

[2] The features of Ruby, Michael Neumann, `http://www.ntecs.de/old-hp/s-direktnet/ruby_en.html`

[3] Ruby core classes, Addison Wesley Longman, `http://www.rubycentral.com/ref/`

[4] Singleton methods, ,`http://www.ruby-doc.org/docs/ruby-doc-bundle/UsersGuide/rg/singletonmethods.html`

[5] Programming Language comparison, Jason Voegele, `http://www.jvoegele.com/software/langcomp.html`