Desirable characteristics(RUBY)

Provability

It does not include any system to write a program and also it’s proof of correctness.

It is possible to use unit tests. This is done by series of assertion methods, e.g.:

**class MyTest <** Test**::**Unit**::**TestCase

**def test1**

t **=** TestClass**.**new**(**10**)**

assert\_equal**(**100**,** t**.**getVal**)**

**end**

The created test will fail, if the result of t.getVal is not equal to 100

Generality

The program is the more general the less different syntactic constructs it allows to use to perform the same functionality [11].

There can be found examples that show Ruby as a general language. An Array is a general object. It has methods **:pop** and **:push** (for adding and removing from the end of an Array ) and **:shift** and **:unshift** (for adding and removing from the beginning of an Array). Therefore, besides the traditional functions of an array, it can be used as a Stack or a Queue. [12]

However some collections have two different methods that perform exactly the same thing – e.g. arrays have synonymic methods size and length.

Extensibility

Ruby is a very flexible language. The user has the freedom to alter even essential and core classes and operators. Even the name of mathematical operators used for number operations can be redefined, e.g. [2]:

**class Numeric**

**def plus(**x**)**

**self.+(**x**)**

**end**

**end**

Now, for adding two numbers, we can call: y **=** 5.plus 6

Essential characteristics

Reliability

It is a dynamically-typed language, which means that as many operations as possible are resolved at the run-time, e.g. [6]:

 type of a variables – one variable name can be used in the code with different data types

 libraries needed to be used together with our code

 order of method calling

It is possible to modify the code during its execution – to enter some new lines of the code, which will execute even if we don’t restart the program run (with the MRI interpreter) [4].

It has a built-in garbage collector, so the memory is freed automatically [4].

Exceptions are handled with help of Exception class (or it’s descendants) [4]: begin # Some code which may cause an exception rescue <Exception Class> # Code to recover from the exception else # optional section - executes if no exception occurs ensure # optional section - always executes, useful e.g. for closing a file end

Orthogonality

„Orthogonality means that features can be used in any combination, that the combinations all make sense, and that the meaning of a given feature is *consistent*, regardless of the other features with which it is combined.“

The author of Ruby thinks that allowing to use any combination of a small feature or syntax leads to complexity of the language. The user can be confused and needs to think like a compiler to know, which syntactic combination is allowed. At one time, one functionality should be used.

However, there can be found examples of orthogonality. On collections (e.g. arrays and hash lists) can be performed the same types of iterations – each and collect methods.

Machine independence

Ruby is interpreted and a cross-platform language. The code written in Ruby can be run on Windows, Linux or Mac, if there is a platform-dependent Ruby interpreter provided.

**Efficient object code**

The YARV virtual machine uses instructions in VMDL (virtual machine description language), which can generate optimized code automatically. For example it includes special instructions for adding two integer numbers. This is useful, because every number is an object and the mathematical operator is in fact method of the number class (and can be redefined). When it encounters the + method and finds out it was not redefined, the special instruction is called and the adding of two objects is more effective. Otherwise more complicated method dispatch will be carried out. [7]

The AOT compilation (Ahead-of-Time) is also available. The Ruby code is firstly translated into a C code and then executed on the YARV, which is more effective than YARV instruction code.