Market Requirements Document

Feature Name: Objectivity/Ruby API

Version: 2	**Date Submitted:** 10/10/07	**Completed By:** Leon Guzenda and Dan Hall
Version: 1	Date Submitted: 06/18/07	Completed By: Leon Guzenda and Todd Stavish

Description of the Problem

Background

"**Ruby** is a reflective, dynamic, object-oriented programming language. It combines syntax inspired by Perl with Smalltalk-like object-oriented features, and also shares some features with Python, Lisp, Dylan, and CLU. Ruby is a single-pass interpreted language. Its official implementation is free software written in C.

Ruby is object-oriented: every data type is an object, including even classes and types that many other languages designate as primitives (such as integers, booleans, and "nil"). Every function is a method. Named values (variables) always designate references to objects, not the objects themselves. Ruby supports inheritance with dynamic dispatch, mixins and singleton methods (belonging to, and defined for, a single instance rather than being defined on the class). Though Ruby does not support multiple inheritance, classes can import modules as mixins. Procedural syntax is supported, but all methods defined outside of the scope of a particular object are actually methods of the Object class. Since this class is parent to every other class, the changes become visible to all classes and objects.

Ruby has been described as a multi-paradigm programming language: it allows you to program procedurally (defining functions/variables outside classes makes them part of the root, 'self' Object), with object orientation (everything is an object) or functionally (it has anonymous functions, closures, and continuations; statements all have values, and functions return the last evaluation). It has support for introspection, reflection and metaprogramming, as well as support for threads[4]. Ruby features dynamic typing, and supports parametric polymorphism.

According to the Ruby FAQ [5], "If you like Perl, you will like Ruby and be right at home with its syntax. If you like Smalltalk, you will like Ruby and be right at home with its semantics. If you like Python, you may or may not be put off by the huge difference in design philosophy between Python and Ruby/Perl." [6]

– Wikipedia 06/18/07.

Problem

It is possible to wrapper both C++ and Java inside of Ruby objects. However, this technique adds an extra mapping layer, making development, debugging and support more difficult and sacrificing performance.

Description of the Requested Feature

1. Objectivity/Ruby - a fully featured Ruby language binding for Objectivity/DB.
2. A persistent version of the most commonly used Ruby standard libraries, such as Rails.
3. A implementation of the Ruby interactive command line interpreter (irb) linked with Objectivity/Ruby.

Part of an existing feature or does it require another feature, if so, which one?

1. New features – Objectivity/Ruby, standard library and irb.

How is this problem being solved now, and why isn't that acceptable?

See "Problem" above.

What languages must support this capability?

- Objectivity/Ruby is a new language binding.
- It must inter-operate with other Objectivity APIs. Object data manipulated by any language must be directly accessible by other languages. The problem of invoking appropriate methods is left to the developer.

Which platforms must be supported?

- At least Linux and Windows (XP or Vista.)

Do any competitors already have this feature?

- Orient claims to support many of the concepts of Ruby, but does not appear to have a direct binding.

Customers who require this feature

- We have requested feedback from our customers via the Objectivity Expert blog.
- TRS Consulting

Revenue at risk, or which could be won

- We are not aware of any business lost as a result of not supporting Ruby, but lack of an interface automatically excludes us from many new projects that have chosen Ruby. It was almost an issue at WebLoq, but they decided to switch to Java.

When is this required?

- Post Release 10.

Additional Notes

We will also need:

- Marketing collateral, including promotional material and a special area on our web site.
- Technical Publications.
- New QA material to prove that the API works and is interoperable with other platform and language combinations.

  2. Licensing costs are to be determined.

Notes

1. Official Ruby site - http://www.ruby-lang.org/en/.

2. We should investigate Ruby/DBI to determine its degree of applicability.

3. We should look carefully at the implications of the reflection feature that Ruby implements, with particular regard to schema evolution and object migration.

"Reflective programming is a programming paradigm, used as an extension to the object-oriented programming paradigm, to add self-optimization to application programs, and to improve their flexibility. In this paradigm, computation is equated not with a program but with execution of a program. Other imperative approaches, such as procedural or object-oriented paradigm, specify that there is a pre-determined sequence of operations (function or method calls), that modify any data or object they are given. In contrast, the reflective paradigm states that the sequence of operations won't be decided at compile time, rather the flow of sequence will be decided dynamically, based on the data that needs to be operated upon, and what operation needs to be performed. The program will only code the sequence of how to identify the data and how to decide which operation to perform.

Reflection can be used for self-optimization or self-modification of a program. A reflective sub-component of a program will monitor the execution of a program and will optimize or modify itself according to the function the program is solving. This is done by modifying the program's own memory area, where the code is stored.
Reflection can also be used to adapt a given system dynamically to different situations. Consider, for example, an application that uses some class X to communicate with some service. Now suppose it needed to communicate with a different service, via a different class Y, which has different method names. If the method names were hard coded into the application, it would need to be rewritten, but if it used reflection this could be avoided. Using reflection, the application would have a knowledge about the methods in class X. And class X could be designed to provide information regarding which method is being used for what purpose. The application, depending on what it has to do, would select the required method and use it. Now, when the different service is being used, via class Y, the application would search the methods in the new class to find the required methods and use them. No modification of the code is necessary. Even the class name need not be hard coded, rather it can be stored in a configuration file, it will be correctly searched for and loaded at run time.

A language supporting reflection provides a number of features available at runtime that would otherwise be very obscure or impossible to accomplish in a lower-level language. These include the ability to:

- Discover and modify source code constructions (such as code blocks, classes, methods, protocols, etc.) as first-class objects at runtime.
- Convert a string matching the symbolic name of a class or function into a reference to or invocation of that class or function.
- Evaluate a string as if it were a source code statement at runtime."

- *Wikipedia 06/18/07.*