# プログラミング言語 Ruby の世界普及戦略

まつもと ゆきひろ（Ruby アソシエーション）

**概要** 日本はソフトウェアが弱いと言われて久しく，実際，日本人によって開発されたソフトウェアで，海外における知名度が高いものは，商用・オープンソースを問わず，ほとんど見当たらないのが実情である．そんな中にあって，筆者によって開発されたプログラミング言語 Ruby は，数少ない例外である．現在，インターネットにおけるプログラミング言語の言及率に基づくプログラミング言語ランキング TIOBE Index で 2010 年 11 月現在 10 位であり，世界中で 100 万人を越えるユーザがいると推測されている．本論文では，Ruby の開発当初から現在にいたるまでの発展段階を俯瞰しながら，将来の「世界に飛び出す日本のソフトウェア」がとるべき世界普及戦略について考察する．

## 1. はじめに

Ruby [1]は，筆者によって 1993 年から開発されているプログラミング言語である．1995 年の公開以来順調にユーザを増やしており，Tiobe Software 社が毎月集計しているプログラミング言語がインターネットでどのくらい言及されているかについてのランキングである Tiobe Index [2]では，2010 年 11 月現在 10 位にランキングされている．2010 年現在，Ruby が国産プログラミング言語，あるいは国産ソフトウェアの中でもっともユーザの多いものであることに異議を唱える人はいないだろう．Ruby の発展の歴史を踏まえつつ，その成功の原因を考えると，以下のような要因が考えられる．

- 優れたソフトウェア

当然ではあるが，ソフトウェアとしての品質が低ければ，世界から相手にされることはない．Ruby はライバルとなる他言語と比較しても十分に優れていたことが，Ruby が世界展開に成功した大きな要因である．また，Ruby がオープンソースソフトウェアであったことも導入に対する障壁を下げている．

- 英語による情報発信

これもまた当然であるが，世界中にいる国産ソフトウェアの仮想ユーザのうち，日本語のドキュメントを読み，かつ理解できる人はごく少数である．我々日本人にとって「英語が苦手」という意識が壁になっているのは事実だが，日本語を理解しない側からこの壁を眺めると，本質的に克服困難である．当然ではあるが，海外に進出するためには，我々の側から心理的障壁を乗り越え，日本人以外にも読める形で情報発信することが必要である．

- コミュニティによるソフトウェアの成長と発展

ソフトウェア，とくにプログラミング言語はただ単にツールとしての価値だけではなく，その周辺に存在する技術や文化などが発展の大きな要因となる．Ruby を周辺としたコミュニティは，プログラミング言語である Ruby の周囲に文化を作り上げ，当初想定した領域をはるかに越えるところまで Ruby を発展させた．

時期的には，「優れたソフトウェア」として Ruby を設計・発展させたのが 1993 年から 1999 年，「英語による情報発信」を行って Ruby の周知が進んだのが 1996 年から 2004 年，コミュニティによる発展が活発になったのが，2001 年から現在までとなっている．これらの発展要因はややオーバーラップしながら，発展の段階を示していると言える．次章以降では，この発展の各段階を個別に見ていくことにする．

## 2. 優れたソフトウェアを設計する

世の中にはまったく革新的なソフトウェアなどほとんどなく，ほとんど全てのソフトウェアには競合する類似ソフトウェアが存在する．プログラミング言語も例外ではない．いや，むしろプログラミング言語はソフトウェアのジャンルとしては特に競合の多い領域であろう．その中にあって「世界に飛び出すソフトウェア」となるためには，なんらかの評価軸で競合に打ち勝つ必要がある．しかし，評価軸は多様であり，必ずしもあらゆる点で競合に勝っている必要はない．目指すべきは「総合的に十分に優れており，それに加えてユーザを獲得できるいくらかの強みがある」状態であろう．

本章では，Ruby を世界的な評価を受けるに足る「優れたソフトウェア」とするために，筆者がどのように Ruby を設計したかについて述べる．

## 2.1 先行言語

筆者は若い頃からプログラミング言語に異常な関心を持っていた．プログラミングに関心を持ち始めた高校生時代を振り返るにプログラミングによって何を実現するかよりも，どのように記述するかという記法に対する関心の方が強く，その記法をデザインし，ひいてはプログラマの思考を支援する，あるいは言い方を変えればある種，思考をコントロールすることに非常な魅力を感じていたように思う．その異常な関心のため，筆者は既存の言語について数多く学んできた．Ruby の設計にあたってはそれらの知識を最大限に活用した．つまり，ライバル（先行者）に勝つためにライバルを最大限に研究したということである．Ruby を設計するために参考にした言語は数知れないが，大きな影響を受けた言語は以下のものである：

- Lisp,
- Smalltalk,
- C,
- Perl.

この中でも Lisp には大きな影響を受けた．高校生時代，ろくなコンピュータ環境に接する機会のなかった筆者にとって，Lisp は文献で読むだけの高嶺の花であり，だからこそ憧れた存在であった．Lisp からは，オブジェクトモデルや，メタプログラミング，高階関数，また Flavors などのオブジェクト指向システムから Mix-in の考え方などを継承している．一方，Lisp 最大の特徴であると言っても良い，S 式やマクロなどは採用していない．これは筆者が S 式とマクロの真のパワーを理解していない「にわか Lisp ファン」であったからかもしれず，あるいは他の多くの非 Lisp プログラマと同様，大量の括弧を忌避したからかもしれない．しかし，Ruby の人気の一部はこの「括弧のない Lisp（つぽさ）」にあるように思われるのは皮肉なことである．

Ruby は，近代的オブジェクト指向言語の祖としての Smalltalk からも大きな影響を受けている．特に単一継承をベースにしたオブジェクト指向システムや，一部のクラスライブラリなどは Smalltlak から直接継承している．一方，Smalltalk のシンプルな文法は受け継いでいない．これは，筆者の文法に対するこだわりを反映しているのだが，結果的に Smalltalk が一般ウケしていない原因のひとつを回避している結果となっている．また，Smalltalk の誇るクラスブラウザを捨てている点は賛否両論あるが，ファイルベースの気軽な開発という点ではメリットがあったと思われる．

Ruby は，C あるいはそれが体現する UNIX 文化を色濃く反映している．当初から UNIX 上で開発された Ruby の多くのメソッドは UNIX のシステムコールや C の標準ライブラリ関数と同一の名称を持ち，プログラミング言語から見て外界である OS との界面は UNIX の影響を強く受けている．最後に Perl [3]であるが，プログラミング言語が対象とする領域において Ruby と Perl は非常にオーバーラップしている．これは，元来 Perl がカバーする対象領域が UNIX プログラマが日常的に直面するテキスト処理をはじめとするスクリプティング領域であり，当初 Ruby を開発しようとした動機のひとつが，そのような日常的タスクを Perl よりはマシなプログラミング言語で行いたいというものであったからである．事実，Ruby という名前も Perl（真珠 Pearl と同発音）の後に続く言語をイメージして名付けたものである．Ruby の当初の設計方針は「Perl ができることができる汎用プログラミング言語を Lisp や Smalltalk を参考にして作り上げる」というものであった．後でも述べるが，この設計方針を反映して，Ruby は一部のユーザに「Better Perl」として受け入れられ，普及の最初のきっかけになっている．

## 2.2 ライバル

1993 年当時の Ruby と類似の対象領域を持つ言語としては以下のものがあった：

- AWK,
- Perl,
- Python.

UNIX 系フィルタ記述言語である AWK は，それ単体というよりはシェルスクリプトの一部として組み込まれることで，テキスト処理を実行するための言語である．当時の UNIX 文化は，シェルスクリプトをグルー（糊）として，単機能のツールを組み合わせてフィルタを作ることで，テキスト処理を行うやり方が主流であった．AWK が提供する単機能によるシンプルさに代えて，Ruby では単一の言語が提供する機能の豊富さで対抗した．幸いなことに，同様のアプローチは Perl によって行われ，すでにかなりの成功を収めていたので，この戦略は Ruby にとっても有利に働いた．

1986 年に登場した Perl は，Ruby の先行言語として参考にさせてもらった．Ruby と Perl を比較したときの最大の違いは，言語としてのクリーンさである．シェルスクリプト的簡易言語から段階的に発展した Perl [4]と比べると，Lisp や Smalltalk のセマンティクスを継承し，汎用言語を意識して設計された Ruby では，すべての値は

オブジェクトであり，すべての機能は，Lisp，Smalltalk，Perl の機能を参考にしたスクリプト領域を意識したクラスライブラリとして整理されていた．

当時，Perl のバージョンは 4 で，後にオブジェクト指向機能を備えることになる Perl5 はまだ存在していなかった．Perl5 になっても，もともとオブジェクト指向言語として設計されていない言語に後付けでオブジェクト指向機能を加えたものであるため，統一感および可読性に難があった．1993 年時点で，オブジェクト指向機能を備えるスクリプト言語はまだほとんどなかったので，この領域におけるオブジェクト指向機能の導入は一般的ではなかった．たとえば，Tcl の作者である John Ousterhout は，1998 年の時点でもなお「オブジェクト指向はスクリプト言語にとってあまり意味はない」[5]と述べている．

しかし，筆者はオブジェクト指向の考え方は領域を問わず有効で，スクリプト言語にとっても例外ではないと考えてきた．2010 年現在，スクリプト言語を含むほぼあらゆる言語がなんらかの形でオブジェクト指向機能を持ち，適応が遅れた Tcl のような言語があまり使われなくなった現状を見ると，1993 年の筆者の判断が正しかったことがわかる．

スクリプト言語の領域にクリーンなオブジェクト指向言語をもたらすという Ruby の基本的設計戦略と，ほぼ同様の方針を持っていたのが Python [6]である．1990 年に登場した Python は，スクリプティングを意識しつつもモジュールシステムなどを持つ汎用言語であり，オブジェクト指向機能も備えていた．スクリプティング領域で，Perl より優れた言語を求めるプログラマたちにとって Python は救済であると考えられていた．

Python と Ruby を比較したときの最大の相違点は，UNIX 的文化との距離である．Python 作者の Guido van Rossum は，当時 Amoeba OS プロジェクトのエンジニアであり，最初のバージョンが Apple 社 Macintosh コンピュータ上で開発された[7]ことからもわかるように，Python は完全には UNIX 的ではない．スクリプト言語であるというよりは，スクリプト言語としても使える汎用言語であることを目指しているように思われた．これはたとえば正規表現機能が言語組み込みでないとか，システムコールなどの呼び出しにもモジュールをインポートする必要があるなどの Python の特性から推察される．一方，Ruby はまったく逆，すなわち汎用言語としても使えるスクリプト言語を目指していた．汎用言語が要求されるある程度以上の規模のプログラムでは，Python の特質は決して不利にならない．しかし，小規模なプログラムでは微妙な違いによる影響が大きくなりがちである．ただし，Python では制御構造をインデントによって表現するという文法上の特徴があり，このことによって行数が削減できるというメリットもある．

Python と Ruby との相違点としては，他にもオブジェクト指向に対する考え方がある．Ruby よりも先行する言語であった Python はやはりオブジェクト指向を重要視しないという先入観から完全に自由ではなかった．スクリプト言語としてオブジェクト指向を備える Python は 1990 年に登場した言語としては画期的ではあったが，それでもなお，オブジェクト指向機能はあくまでもオプションで，後付けのものにすぎなかった．言語が提供する基本データはオブジェクトではなく，各ライブラリもクラスライブラリではなかった．最初からオブジェクト指向言語として設計され，ほとんどの機能がクラスライブラリによって提供されている Ruby は，この点において Python よりもはるかに統一感があり，高い記述力を持っていた．

しかし，これらの言語と比較して Ruby がいかに優れていたとしても，1993 年の時点，あるいはインターネットを通じて公開した 1995 年の時点では Ruby の知名度はほぼゼロであり，いまだ誰も知らないプログラミング言語でしかなかった．

## 3. 世界に飛び出すための取り組み

### 3.1 英文ドキュメントの効用

日本語で情報発信している限り，受信してくれるのは日本人，あるいはごくまれに存在するかもしれない日本語が読み書きできる非日本人だけである．それでもよいという判断もありえるが，筆者はそれを望まなかった．Ruby の有用性は日本・日本人に固有のものであるとは思えなかったためである．そこでインターネットに公開した直後の 1996 年ごろから Ruby のリファレンスマニュアルの英訳に取り掛かった．

筆者がドキュメントの英訳にこだわったのには理由がある．筆者は昔から各種フリーソフトウェア，特に言語処理系のソースコードを読むことを趣味としていたのだが，ある時，大変興味深い実装ではあるが，ドキュメントやコメント，関数名が英語ではない読めない言語で記述されるソースコードに遭遇した．なにぶん 20 年近く昔のことでその言語がなんであったか，記憶はすでに曖昧であるが，イタリア語かロシア語であったのではないかと思う．結局その言語処理系の読解はあきらめてしまったのだが，その経験以来，我々日本人が日本語でドキ

ュメントやコメントを書き，関数名・変数名にローマ字を用いる時，それは日本人以外の人に私と同じ思いをさせることになる，それは大変もったいない，と強く感じるようになった．

正直，筆者は英作文が得意ではなく，ドキュメントの英訳も適切な訳が思いつかず，業務の間に，インターネットで類似の表現を探してきて切り貼りするような情けない作業を泣きながら継続することにより，半年程度でドキュメントの翻訳作業はなんとか完了した．表現は拙く，記述は十分ではなかったが，少なくともなんとか英語として読めるドキュメントが登場したことにより，日本語を解さない人であっても，この東洋からの未知の言語について学ぶことが可能になった．

ドキュメント英語化の効果は大きく，1996 年中には海外からの反応をいくつかいただき，日本語圏以外のユーザからの強い要望に従い，1998 年には英語によるメーリングリストが開設されている．

その後の展開は急激である．1999 年末には Pragmatic Programmers を名乗るソフトウェアコンサルタント（Dave Thomas と Andy Hunt のふたりから構成される．主たる著書は『The Pragmatic Programmer（邦訳:達人プログラマ』[8]) から，Ruby についての書籍を書きたいとコンタクトがあり，2000 年秋には『Programming Ruby: A Pragmatic Programmer's Guide』として出版された[9]．この書籍は後にインターネットで無償で公開され，英語圏での Ruby の啓蒙に大変役立った．

2001 年にはフロリダ州タンパで ACM OOPSLA 2001 の直前の週に，First Annual International Ruby Conference が開催された．この年のカンファレンス出席者は 34 名であった．この時点では毎年開催されることは確定しておらず「First Annual」の部分は一種のシャレであったのだが，その後も開催は継続され，10 回目となる International Ruby Conference 2010 はルイジアナ州ニューオーリンズで開催され，出席者は 850 名を超えた．

2002 年にはわずか 4 年にして英語メーリングリストの累計メール数が 3 年早く開設された日本語メーリングリストのメール数を超え，英語圏での Ruby に対する関心の高さが浮き彫りになった．2001 年の第 1 回 Ruby Conference の時には，Ruby 関連の日本語でしか得られない情報を，機械翻訳なども駆使していかにキャッチアップするかというテーマが真剣に議論されるなど，初期の海外 Ruby コミュニティにおいて日本からの情報をいかに早く入手するかについて真剣に関心が持たれていたこともあったが，わずか数年にして英語での情報が日本語の情報を上回ることになった．これはやはり英語圏（非日本語圏）のエンジニアの層の厚さを示していると考えられる．

## 3.2 Web 領域への進出

2004 年には，Web アプリケーションフレームワークである Ruby on Rails が登場し，爆発的な話題を呼んだ．Web アプリケーションフレームワークとしての Ruby on Rails は，既存のものと比較して，以下の特徴があった：

- コードの重複を忌避する DRY (Don't Repeat Yourself) 原則，
- 設定による柔軟性よりも，実践的なデフォルト値を提供することで設定不要を目指す CoC (Convention over Configuration)，
- 既存のクラスを積極的に拡張することによる Ruby の Web アプリ用 DSL (Domain Specific Language) 化，
- プログラムや設定ファイルによる明示的な指定を削減し，DRY 原則を実現するためのメタプログラミングの活用．

これらにより実現される生産性の高さが話題を呼び，Web 領域での Ruby の存在感が増大した．特に新しいテクノロジーに敏感なシリコンバレーのスタートアップ企業で数多く採用された．たとえば，アメリカ職業別電話帳を提供する Yellowpages.com [10]や，Twitter [11]などが Ruby を採用している．

アメリカなどでの Ruby の成功を受け，2007 年頃から日本でも逆輸入のような形で Ruby の採用が増加した．たとえば，クックパッド[12]や楽天[13]などは比較的早い時期から Ruby を採用している．

## 3.3 Ruby と共存共栄のコミュニティ

Ruby の発展の過程で以下の言語およびその周辺コミュニティの一部が Ruby を受け入れてきた：

- Smalltalk，
- Perl，
- Python，
- PHP．

Smalltalk コミュニティの一部は早期から Ruby に注目していた．初期の Ruby Conference は毎年 OOSPLA の直前に同じ街で開催され，OOPSLA に参加するオブジェクト指向のエキスパートの参加を促していた．特に Smalltalk を起源とするテスト駆動開発コミュニティのメンバーは Ruby にもそれを持ち込んだ．元 Smalltalker の Martin Fowler は 2006 年の段階で「私は Ruby が大好きだ」と述べ，また「生産性が重要であれば，実案件にも Ruby

が使用できる」と述べている[14].

Perl コミュニティもまた Ruby に人材を提供している．初期の Ruby は Perl に大きく影響を受けたため，Perl からの移行は比較的容易であった．

その上 Ruby は Perl5 よりも一貫性の高いオブジェクト指向機能を提供しているため，プログラムの可読性などを実現するため，また，Ruby on Rails を利用するために Perl から Ruby への移行が数多く行われた．

Ruby と Perl の関係はそれだけにとどまらない．Perl の新規再設計となるバージョン Perl6 では Ruby からの影響と思われる機能をいくつか備えている．その中には引数展開や Mix-in がある．

Python は Ruby とライバルとなる言語であり，お互いを意識してきた．しかし，Python の方が先行し，はるかに大きなユーザベースとライブラリの蓄積を持つため，長らくもっぱら Ruby の方が Python を参考にさせていただく関係であった．しかし，Ruby on Rails 登場以後は Python コミュニティも Ruby をある程度は意識しているようである．

PHP [15]は Web アプリケーション開発を対象にしたプログラミング言語である．PHP コミュニティは Ruby コミュニティよりもはるかに大きく，また，一般的に PHP プログラマは Ruby プログラマよりも見つけやすい．このことから，PHP ユーザの Ruby に対する反応は様々である．典型的なものとしては「Ruby の方が技術的に面白いが，PHP には仕事がある」というものである．しかし，Ruby on Rails 登場以後は，ある程度 Ruby への移行も発生しているようだ．また，PHP 側も CakePHP [16]や Symfony [17]のような Ruby on Rails を意識した Web アプリケーションフレームワークを開発している．

## 4. コミュニティによるソフトウェアの成長と発展

1995 年の公開当初から Ruby は自由なライセンスとともに配布された．現在の言葉でいえばオープンソースであった．そのため，ユーザが自由にソースコードを閲覧することが可能であり，バグ報告や修正などの形で開発に参加しやすかった．

そのこともあり，他の多くのオープンソースコミュニティと同様，Ruby においても早くからユーザコミュニティ主導の開発が行われてきた．

最終的な意思決定こそ，開発リーダーである筆者によって行われるものの，様々なプラットフォームへの移植，明らかなバグ修正などは，筆者の判断を待たず早いペースで行われてきた．また，そのような参加型のコミュニティ姿勢は Ruby そのものの開発を行うメンバーにとどまらず，Ruby を使う全てのユーザに，なにか新しい面白いことを Ruby で行ってみることを促した．そのような開発志向のコミュニティが Ruby の魅力の一部であったことは間違いない．

「新しい面白いものを開発する」という姿勢によって駆動されるコミュニティは人間関係も改善した．Ruby コミュニティ活動は初期の段階においては主にメーリングリストを通じて行われてきたが，参加にあたって審査のない出入り自由なメーリングリストにおいて，時折，否定的な暴言を吐く存在の乱入は避けられない．他のコミュニティもこの問題については悩まされてきた．しかし，Ruby はそのような暴言に対して，より生産的な方向を示唆することで対処してきており，「ナイスなコミュニティ」[18]として知られている．

このような生産的なコミュニティ活動を 15 年に渡って世界的に継続できたことこそが，Ruby 世界発展の秘密ではないかと考える．

## 5. ビジネス領域への発展

2004 年の Ruby on Rails 登場以来，Web 領域における Ruby の存在感は拡大の一途であるが，近年ビジネスソフトウェアのプラットフォームが Web に移行するに伴い，ビジネス領域での Ruby 利用も拡大している．

しかし，ながらくエンジニアの自発的関心を動機づけとしてコミュニティベースで発展してきたオープンソースソフトウェアである Ruby の開発体制は，ビジネスユーザに対して十分な支援を提供できないことがあった．

そこで日本における Ruby のビジネス利用が本格化した 2007 年，Ruby のビジネス利用に伴い発生する課題を緩和するための団体である Ruby アソシエーションを組織した．Ruby アソシエーションでは，

- Ruby 開発の支援，
- Ruby 技術者認定試験の運営，
- Ruby 技術を持つ SI 業者の認定，
- Ruby に関する情報の提供．

などを行う．

さらに 2009 年には Ruby の国際標準規格策定にも力を入れている．これは情報処理推進機構（IPA）が主体となり，Ruby アソシエーションが協力している形で策定されている．2011 年には日本工業規格（JIS）として登録されるべき作業中であり，その後，国際標準化機構（ISO）の標準規格として登録予定である．

国際標準規格化によるメリットとして

- 公共調達の拡大，

- 言語仕様の安定，
- 実装の信頼性向上

などが考えられる．これらのメリットにより，ビジネス領域における Ruby 使用により安定感が増大することが期待される．

特に，実装の信頼性向上についてだが，ここ数年，JVM 上で動作する JRuby，マイクロソフト.NET フレームワーク上で動作する IronRuby，C++で実装された Rubinius，商用 Smalltalk 処理系 GemStone の VM で動作する MagLev など，Ruby 言語の別実装が数多く登場している．標準規格として明確に文書化されることにより，これらの処理系相互の互換性向上および，仕様の安定が期待される．

## 6. まとめ

Ruby は世界中で人気が高いプログラミング言語であり，日本人によって開発されたソフトウェアとしては例外的である．これは既存の言語を十分に研究し，また当時としては比較的先進的な言語機能を使いやすい形でパッケージして，アカデミック領域以外の「普通」のソフトウェア開発者に届けたことが大きいと考えられる．また，当然ではあるが，早い時期から英語のドキュメントを用意し，英語コミュニティが Ruby の発展・普及を駆動してきたことも重要である．

Ruby はオープンソースソフトウェアであるため，Ruby が採用した戦略がすべてのソフトウェアプロダクトに有効であるとは思えないが，しかし，本論が参考になり，より多くのソフトウェアプロダクトが日本・日本語という「見えない壁」を打ち破り，世界進出を果たすことを願ってやまない．

## 参考文献

1) Matsumoto, Y. et al: The Ruby Programming Language, http://www.ruby-lang.org/
2) TIOBE Programming Community Index http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
3) Wall, L. et al: The Perl Programming Language, http://www.perl.org/
4) Wall, L., Christiansen, T. and Schwartz, R.: Programming Perl, Second Edition, O'Reilly and Associates, ISBN 1-56592-149-6 (1996).
5) Ousterhout, J.: Scripting: Higher Level Programming for the 21st Century, IEEE Computer magazine, pp.23-30 (March 1998). http://home.pacbell.net/ouster/scripting.html
6) van Rossum, G. et al: Python Programming Language. http://www.python.org/
7) Lutz, M.: Programming Python, O'Reilly, ISBN 1-56592-197-6 (1996).
8) Hunt, A. and Thomas, D.: The Pragmatic Programmer, Addison Wesley, ISBN 0-201-61622-X (1999).
9) Thomas, D. and Hunt, A.: Programming Ruby: A Pragmatic Programmer's Guide, Adison Wesley, ISBN-0201710897 (2000).
10) Yellowpages.com. http://www.yellowpages.com/
11) Twitter. http://twitter.com/
12) クックパッド. http://cookpad.com
13) 楽天. http://www.rakuten.co.jp
14) Fowler, M.: EvaluatingRuby (2008). http://martinfowler.com/bliki/EvaluatingRuby.html
15) http://www.php.net/
16) http://cakephp.org/
17) http://www.symfony-project.org/
18) Fowler, M.: RubyPeople (2005). http://martinfowler.com/bliki/RubyPeople.html

まつもと　ゆきひろ（正会員）
1990 年筑波大学第三学群情報学類卒業．同年，（株）日本タイムシェア入社．1994 年トヨタケーラム（株）入社．1997 年（株）ネットワーク応用通信研究所入社．2007 年より Ruby アソシエーション理事長，および（株）楽天 技術研究所フェローも兼務．オープンソースソフトウェアの開発に従事．プログラミング言語の設計と実装に興味を持つ．ACM 会員．

投稿受付：2010 年 12 月 24 日
採録決定：2011 年 2 月 16 日
編集担当：竹内　郁雄（早稲田大学）
　　大蒔　和仁（東洋大学）