Rubyist Magazine 巻頭言選集

高橋征義 著

2013-03-20 版　　達人出版会　発行

はじめに

(読者が読みたくなるようなまえがきっぽい言葉)

内容について

(内容紹介)

動作環境について

(バージョンとか)

謝辞

(必要に応じて)

目次

第1章

「Ruby らしさ」について

最近気になっている言葉がある。「Ruby らしさ」という言葉だ。この言葉について、いくつかの角度から考えてみたい。

「Ruby らしさ」ということが気になったのは、Ruby 1.8.0 のリリース前のころだろうか。Ruby は 1.8 系から、まつもとさん以外の方が作られたライブラリを 積極的に標準添付するようになった。このため、多くのライブラリが別途インストールする必要なく 使えるようになり、利便性は非常に高まったのだが、いくつかのライブラリに関しては「Ruby らしくない」という評価を聞いた。具体的には、メソッドの名前付けや挙動が Ruby の標準的なクラスのそれとずれがある、といった指摘である。こうなると、Ruby を使う上での直感が効きにくくなるため、プログラミングにも影響を及ぼす。しかし、このような「Ruby らしさ」は明文化されていないため、共有がしづらい。そのため、新規で作られるライブラリが Ruby らしくなることはあまり期待できない。しかし、Ruby の標準になるライブラリはこのように Ruby のほかのライブラリとは 独立した作者によって独立に作られることが多いので、これでは「Ruby らしさ」が徐々に失われていく危険性がある。

これとは少し異なるが、「Ruby らしさ」という言葉は、最近出版された「プログラミングのための線形代数」[*1]でも使われていた。こちらの方は、「Ruby らしくない」という言葉が、他の言語（特に非オブジェクト指向言語）で書くのと同じような書き方をする、といった意味で使われていた。

しかし、よくよく考えてみると、このような目的で「Ruby らしい」プログラムを書くことは、どの程度問題があるのだろうか。そして、「Ruby らしくない」Ruby のプログラムは、どれくらい分かりやすくなるのだろうか。そして、こういった「Ruby らしい」書き方は、どういった内容が失われたり、捻じ曲げられたり してしまうのだろうか。

Ruby を目標としていない目的で Ruby で書く際、「Ruby らしく」書けない、ということは、「Ruby らしさ」に何かしらの問題があることを暗示しているような気がしてな

[*1] http://www.amazon.co.jp/dp/4274065782

らない。 それとも、Ruby らしく書かないというのは、 数日ないし数回の授業分の講義期間といった短期的にはともかく、 中・長期的には特に効果がない戦略であり、 通常は Ruby らしく書いたとしても 何ら問題がないのだろうか。

似たような話題で、先日まつもとさんの日記上で行われた議論では、 Ruby と HSP との比較において、 Ruby よりも HSP の方が文法要素がシンプルであるため、 習得しやすいのではないか、という意見があった。 これは要するに「Scheme は構文は S 式しかなく、仕様自体も 数十ページでおさまる程度しかないので習得しやすい」 といった意見と同型のものかと思われる。 ここでは HSP の「シンプル」な文法要素についてと Scheme の それを比較することはしないが、 「Ruby らしさ」の追求が何かしらの敷居をもたらすのは あまり歓迎できる事態ではない。この敷居は本質的なものであり、 「Ruby らしさ」は習得にコストがかかるものなのだろうか。 それとも、習得のコストは何かしらの形で軽減が可能であり、 そのような道を模索するべきなのだろうか。

さらに、Ruby Conference 2004 での Nathaniel さんの 発表も思い出してしまう。彼は発表の中で、現在の Test::Unit が 「Ruby らしくない」といって批判し、より「Ruby らしく」した test/unit2 を開発中だそうだ。そのサンプルの一部は 発表の中で公表されていたが、クラスではなくブロックを多用する 形に大きく変わっていた。これはつまり、Ruby での プログラミングは、挙動によって多数のクラスを使い分ける ようなものではなく、挙動の違いをブロックで指定できるように なる、ということかもしれない。これは OOP としては 若干特殊な戦略かもしれない。 とはいえ、これはこれで 「Ruby でできる (Ruby 以外の言語ではできないか、しづらい) こと」と 「Ruby らしいこと」を混同しかねないのではないか、という懸念もある。 実際、サンプルコードには違和感があった。この問題は上に書いた標準添付の問題に結びつくものでもありそうだ。

あなたの書いている Ruby のプログラムは「Ruby らしい」プログラムだろうか。 それとも「Ruby らしくない」プログラムだろうか。 その「Ruby らしさ」あるいは「Ruby らしくなさ」は、 あなたのプログラムにどのような影響をもたらしているのだろうか。 あなたのプログラミングの何を促進し、何を阻害しているのだろうか。 そしてあなたのプログラムは、 「Ruby らしさ」の何を享受し、何を排斥しているのだろうか。 「Ruby らしさ」、そして Ruby にどのような影響をもたらしているのだろうか。

第2章

コミュニティについて

> 「そんなことでは地球は滅びてしまうぞ」彼はそんな非難に対して このようにこたえる男だった。「それがどうした」
> (神林長平『騎士の価値を問うな』(ハヤカワ文庫 JA『戦闘妖精雪風』所収) より)

先日行われた Open Source Way 2004 で、 佐渡秀治氏による『日本におけるオープンソースの幻想と VA Linux』 という発表があった。ITmedia や japan.linux.com といったニュースサイトで 取り上げられ、スラッシュドットジャパンでも 300 を超えるコメントがつく 大きなスレッドに成長したこともあり、ご存知の方も多いだろう。

この話題、特に後半のコミュニティ活動についての批判的な言及は、 「○○ユーザ会」に近いような団体を遅まきながら 立ち上げた私としても、非常に気になっている。 あるいは私も、せっかく世界に向けて活躍をしていた Ruby を、 日本 (語) の中でタコツボに押し込めるべく、 利権を求めて活動しているワルモノ (ただし小物) のように 見らなくもないかもしれない。 実際、「利権」については設立の際に考えないわけでもなかった。 といっても、技術力も英語力も発言力もたいして持っていない私が 利権を享受できないのは明らかで、 むしろ利権を求めて怪しげな人に介入され、 会の活動全体がおかしくなる危険はないか、 などといった点を気にしていたのだが。

実を言うと、佐渡氏の現状認識そのものにはあまり異論はない。 「日本 Ruby の会」が「日本 Ruby ユーザ会」という名前になっていないところや、 活動方針に「ユーザの支援」だけではなく 「開発者の支援」と明記しているところからも察していただける通り、 「ユーザ会」という組織を運用していくことはなかなか難しいものだと考えている。他の団体についてはあまり詳しく内情を知らないのだが、 問題のある団体もきっとあるだろう。

しかしながら、その対策、結論についてはにわかに首肯できない。

そもそも、オープンソースには、「開発元」に対する「貢献」が必要なのだろうか。 やりたい人がやりたいことをやりたいようにやる。 それで十分ではないか……といった意見がスラッシュドットに書かれていたが、 同感である。 「そんなことではオープンソー

スは滅びてしまうぞ」と言われて「それがどうした」と答えるような人を許容することができなくなって しまうのは、狭量というものだろう。

「ローカルでパッチを作りつづけるよりも本家に還元した方がコストが低い」 という指摘もあった。おおむね正しいと思うが、開発元に理解がない場合などは、 その労力が馬鹿にならない場合もあるのではないか。 諸事情によりそのように状態に陥っている場合、それを非難したところで 得られるものは少ないだろう。

究極の軽さと WYSIWYG を追求したプレゼンテーション手法である「less プレゼン」などの魅力的なネタで知られる塩崎拓也氏は、 以前「オープンソース」の「貢献」に対し「おすそ分け」という 概念を対峙させていたことがある。「つまらないものですが」と恐縮しながら さりげなくパッチを差し出す姿を想像すると、 なにやら妙に日本的な情緒を醸し出しているようでもあり、 このようなメンタリティには強い共感をおぼえる。「おすそ分け」と「貢献」。 行為そのものは同じようなものに見えても、そこに働く「気持ち」は大きく異なる。

「おすそ分け」はもろく、弱い。 「おすそ分け」程度の気持ちでは、理解のないかもしれない開発元に対し、 その意義を手を変え品を変え説明し説得する、 といった活動を支えきれるほどの強い動機にはならないかもしれない。 いや、そもそも働きかける以前に、批判されることを恐れて 何もできなくなってしまうかもしれない。 その場合、その努力はあっさりと狭い「タコツボ」の中でしか 享受できないことになってしまう。 それでも、そのときはそのときで、 説得できる材料と人材がそこに現れることを気長に待つ、 という選択肢もあるだろう。それを批判し、 アジテーションを投下しても、批判された側の人が 突然貢献に目覚めて活動をはじめる、 といった効果はとてもではないが期待できない。 (アジテーションには議論を盛り上げるという効果があり、 この発表やその紹介記事もまさにその成果と言えるのかもしれないが、 手法としては決して誉められるものではない。)

もちろん、何もせずに待ち続けていなければいけない、というわけではない。 季節の挨拶も通じなさそうな開発元にいきなり働きかけることはできなくとも、 周りの人には細々と働きかけることならできる、というのであれば、 ぜひそのようにするべきである。そうすれば、いつか時と人がそろい、状況が変わることもあるかもしれない。 あるいは、自分が変わり、いろいろな方の助力により、 開発元に対して声が届くようになるかもしれない。 そのためには、それがたとえ「支流」の中だけであったとしても、 働きかけをしてみることである。その小さな一歩を大きな流れへと 広げていくことは、まさにコミュ

ニティがなしうることだろう。

Rubyist Magazine は、今のところ日本語でしか提供されていない。 その意味で、日本 (語) に閉じた成果物といっていいだろう。 けれども、現在までの Rubyist Magazine の誌面には、 Ruby を取り巻く魅力が感じられるように思う。 実際のところ、私が直接貢献しているものは何もないに等しいのだが、 誌面全体からは「Ruby を使うのが好き」「Ruby を使うのがたのしい」 というメッセージが伝わってくる。 いまのところ本誌からは、執筆者にも編集者にも、 何の謝礼も提供されていないのだが、 報酬もなしに書き紡がれていく記事たちは、 まさに「たのしさのおすそ分け」でもあるように思う。

記事の中には専門的なものや技術的に高度なものもあり、 多少 Ruby をかじっただけの読者でも理解できない記事もあるだろう。 しかし、それでも「なんだかよくわからないけど、このひとたち楽しそうだね」 くらいは思ってもらえるかもしれない。 そうすれば Ruby にも好意を持ってもらえるだろうし、 自分も活動にコミットしようと思うひともいるだろう。 それが明日の Ruby をさらに魅力的なものにしていくのではないか。 こういった伝播力は「おすそ分け」の強みである。日本の社会のなかで 「おすそ分け」という言葉と風習が根強く残っているのも、 社会システムの中で効果的な役割を担うほどには強力な機構だったためだろう。

> 遠くにいるその人に語りかけるように、すぐそばにいる誰かに 笑顔で語りかけてあげてください。それは、別々のことではありません。
> (「谷山浩子・猫森集会 2003」コンサートパンフレットより)

現在、Rubyist Magazine では、非日本語圏の方に執筆を依頼する準備を進めている。すでに Ruby Conference でのメッセージや、RLR でのライブラリ作者の コメントなどで、海外の方の声を寄せていただいたことはあるが、 原稿依頼の交渉をされている担当者のメールを読むと、 それとはまた別の苦労があるように見える。 それでも、担当者の尽力もあり、実現に向けて着実に歩を進めている。 また、Rubyzine という英語圏での Ruby 雑誌が創刊されれば、 記事を翻訳し掲載したい、という話もある。

もともと日本語圏の方々のための、 日本で閉じたものとして作ってきた本誌でさえも、海外へとつながっていく道がある。 私たちの隣にいる人と、違う国・違う言葉・違う風習・違う思想で 暮らしている人とは、どこかでつながっている。

「世界に目を向ける」ように、自分の身近にいる人にも目を向け、 また身近な人に目を向けるように、遠い世界にいる人のことを思う。 そんな風にしていければいいと思う。

それらはみな、別々のことではないはずなのだから。

第3章

誰のための Ruby?

前号発行からの短い間にも、 海外での Ruby の注目度がより高まってきたように感じている。

本誌でももりきゅうさんの連載で扱っている、 以前から次第に注目を集めていた Web アプリケーションフレームワーク、 Ruby on Rails (RoR) の 紹介記事がオライリーのサイトに掲載された (Rolling with Ruby on Rails)。 「Java のフレームワークよりも 10 倍早く開発できる」という煽り気味の宣伝文句と共に 公開されたその記事は、 Java や Python のユーザを巻き込んで多くのサイトで Ruby と RoR が取り上げられたようだ。 RoR の blog では、いささか過剰ではないかと思われるほどに賛否両論の反応を (そもそも「賛」なのか「否」なのかもわからない、 まつもとさんによる日本語の日記の記事のキャプチャすらも) 取り上げ、 自身のサイトとアプリケーションの盛り上げに貢献させている。 そのあおりを受け、Ruby のインストーラのダウンロード数も増加したり、ruby-talk の流量も 4000 件を軽く越え、過去最高を記録したりと、 影響力はあなどれない。

また、why the lucky stiff さんの blog サイト、redhanded では、 日本の Ruby 界隈の情報を紹介する人を募集し、 それに応えてはんばあぐさんと馬場さんが ゲストブロガーとして登場するようになったのも見逃せない。 新年会や Debian の話題をとりあげるはんばあぐさんや、 Python の型付けに関する Matz にっきでの発言を翻訳する 馬場さんのコメントは、日本からの情報に飢えている 海外の Rubyist からの歓迎と感謝の言葉が相次いでいた。

海外で Ruby に近いポジションで使われている言語は何と言っても Python だろう。 かの Bruce Eckel 氏も、 「Ruby が現れるまで、Python の対抗馬は現れなかった。 そして今は、そのライバルがいる」と言っている (Re: Fixes What's Wrong With Python)。 その勢いはまだまだこれからも伸び続けていきそうである。

とはいえ、Ruby の現状を手放しでは喜べるわけではない。Ruby に対する辛い評価もある。

昨年暮れ、めでたくリリースされた Ruby 1.8.2 だが、 そのリリースエンジニアリング

については、 パッケージングに失敗したり、 更新内容の告知などの点で、 まだまだ改善の余地があることを思わされた。 もっとも、私自身は開発には何の役にも立たないので邪魔にならないよう周りをうろちょろしてたまに突いてみたりする程度だが、 どうせ前日になって突付くのであればもっと早めにつついておけばよかった、と反省している。

なお、更新情報については、 sheepman さんなどの手によって書かれた (らしい。Wiki なので詳細は不明であるが) 1.8.2 の更新情報のページの形で補完されている。ありがたいことである。

また、これに関連して最近 ruby-dev ML で話題になったのは、1.8.2 の互換性についてである。 Ruby では、1.8.0 と 1.8.1、1.8.1 と 1.8.2 など、 3 つ目の数字が変わっただけのバージョン違いのものでも、 挙動が変わることがある。1.8.1 と 1.8.2 についても上記の更新内容のところで紹介されているが、 これまでのところあまりはっきりとした基準のないまま変更の必要性が判断され、 実施されてきた。

更新内容の公開にせよ、バージョン間の互換性にせよ、これらの影響を大きく受けるのは、 Ruby そのものの開発にはコミットしていない Ruby ユーザである。 つまり、自分で Ruby のアプリケーションを作成しているユーザである 「Ruby アプリケーション作者」と、 誰かが作成した Ruby のアプリケーションをそのまま使用する 「Ruby アプリケーションユーザ」である。

彼らには強く Ruby に関わる動機も理由もない。前者のアプリケーション 作者にとってみれば、自分が使いやすいように Ruby が変更される ことは歓迎だが、そのための説得にわざわざコストを費やそうとか、 高くもあり低くもある「敷居」を越えようとはなかなか思わないだろう。 また、Ruby に余計な変更が加われば、今まで自分が作成したアプリケーションに 手を入れなければならなくなる必要がある。バージョン間の互換性が 問題になるのであれば、バージョンを判別しふるまいを変更する 特別なコードを新たに追加しなければいけなくなるかも知れない。 それは厄介である。 さらに後者の、Ruby アプリケーションを利用するユーザにとってみれば、 Ruby そのものが改良されようがされまいが意味がない。 Ruby 1.8.1 用に作られたアプリケーションを使いたいユーザにとって、 そのソースが正常に実行できなくなってしまった Ruby 1.8.x は、 単に「壊れた Ruby」でしかない。

何かしら Ruby が「良く」なったとしても、 彼ら Ruby ユーザにとってそれが本当に「良い」ものだと 認識されているかどうかは疑問である。 彼らにしてみれば、 多少言語が「良く」ならなくとも、 以前のバグを含めた互換性が確保されるほうがよほど望まし

いし、 逆に変更しなければいけない点はあらかじめ十分に告知されていなければならない。 それができないなら、サポートの手間が増えるだけであり、 いっそリリースされない方がよい。 そう考えていても不思議ではない。

……と、このように書けば、前号での巻頭言の言葉も踏まえ、「私たちはもっと見知らぬユーザのことを考えなければならないのだ」 といったような流れになると思われるかもしれない。 しかし、話はそう簡単ではない。

ここで Linux Magazine に掲載されていたまつもとさんの連載から、 2003 年 2 月号の記事を引用したい。

ああ、重要な点を忘れていました。一番大切にしなければならないのは 開発コミュニティであってユーザーコミュニティではない、ということです。 少々暴論ですが、フリーソフトウェアを開発する動機でもっとも大きいものは、 「ソフトウェア開発したいから、楽しいから」と言うものだと思いますが、 単なるソフトウェアユーザーはなかなかそれを理解してくれません。 フリーソフトウェアの場合、どれだけユーザーがいても「もうけ」が あるわけではありませんから、より楽しい開発に集中するのは むしろ当然だと思います。すぐれたソフトウェアのユーザーコミュニティは、 開発コミュニティの周辺に自然発生すると思います。 (Ruby 開発日記「大人になる、ということ」より)

明確に開発者優位の姿勢をとっていることがわかる。

これはユーザを置き去りにした「わがまま」な態度なのだろうか？　そうではない。言語開発者が開発を中止してしまえば、 いくらユーザが望んだところでその言語は衰退してしまう、 ということは、別に脅しでもなんでもなく、端的な事実である。 個人が開発している言語の寿命は開発者が握ってしまっている。 その意味で、Ruby 開発者が開発者 (コミュニティ) 優先を唱えることには何の齟齬もない。 順調に続いているように見えている Ruby の開発も、 他愛無いことであっさり中断される危険性を孕んでいる。

この事実を持って、Ruby の開発基盤の脆弱性を問題視する向きもあるかもしれない。しかしながら、ほとんどのフリーソフトウェア・オープンソースソフトウェアに よる言語処理系は同様の事情にあるように思う。 ましてや Ruby は「プログラミングのたのしさ」といった、 露骨に主観的な要素を重視する言語である。 開発者の意向をかなりの程度優先させなければ、 開発の持続可能性を確保することは困難だろう。

とはいえ、開発の継続のために、開発者はユーザの意向を軽視してもいいのだろうか？

それもまた極論である。

豊かな開発コミュニティを維持するには、新たにコミュニティに帰属する者が一定数以

上求められる。 以前からの Ruby ユーザであれば、 かつては内外の Ruby 関連メーリングリストなどを舞台に活発な活動をされていたにも関わらず、 久しくご無沙汰になっている方々を幾人も思い出せるだろう。 彼・彼女らに代わって活躍するのは、当時は Ruby の存在そのものも 知らなかったかもしれない新参者たちである。

そのような新しい成員が新しいプロダクトに関わるのは、たいていユーザとしてだろう。 そして、開発者コミュニティを活性化させるような開発者であれば、 そのプロダクト自身の潜在的価値を見抜き、 それがユーザである彼・彼女からも十分魅力的でなければ、それ以上関わろうとしないはずだ。 その意味では、影響力のある開発者を集めるためにも、 その言語がユーザにとっても魅力的なものでなければならない。 すなわち、「開発者優先」というポリシーと、 「ユーザ重視」というポリシーは両立するのではないか。

いろいろ書いてはみたが、 正直な話、私自身、この問題については落とし所が見えていない。 あるいは、ここでも「バランスが重要」ということになるのかもしれない。 見えないユーザ、見えない開発協力者、見えない理想の言語を目指して、 一歩ずつ Ruby の開発が進められていく。 異なる立場の人々によるせめぎあい自体に関与することも、Ruby という言語がもたらすたのしみの一つであるとも言える。 私としては、 このせめぎあいの場所に立ち会えたことに感謝しつつ、 せめて自分なりの貢献を通して、 たのしみの一端を享受できれば、と思っている。

第4章

読み手として、書き手として

仮にあなたが古くから、例えば 1999 年から Ruby を使っていたと仮定しよう。

おそらく、あなたは ruby-list ML を購読していたはずである。 なにせ、まつもとさんによる世界初の Ruby 解説書、 『オブジェクト指向スクリプト言語 Ruby』ですら 発行されたのはその年の秋のことだ。 当時はメーリングリストのメールを読んでいなければ、まともに Ruby の情報を得ることができなかったのだ。

そしてあなたは、きっとこんなメール ([ruby-list:17335] rubyunit) を読んでいたはずである。

> 助田です.
> 突然ですが，Kent Beck の Testing Framework を ご存知でしょうか？ 私は良く知らないのですが，友人から Ruby 版が あるかと聞かれたので友人に内容を確認しつつ 作ってみました. (略)

もちろんあなたはここに書いてあるメールの URL にアクセスするだろう。 実は URL が間違っているのはご愛嬌だが、そこに書かれているサイトの一つ、 その「友人」の方が書かれている 「Kent Beck Testing Framework 入門」というドキュメントを読むことになるだろう。 ……読んでどのように思われるだろうか？ もしかしたら、あなたはその時すでに「デザインパターン」本、 通称 GoF 本のことはもう知っていたかもしれない。けれど、 C++ や Smalltalk のことはあまり知らず、GUI のツールキットの話も ぴんとこなかったのであれば、 これを読んでデザインパターンの応用について、 蒙を啓かれる思いを持ったかもしれない。 そして、Kent Beck の XP 入門本が翻訳される遥か以前に、eXtreme Programming やテスト駆動開発へとつながる 「テストファースト」という手法について、 実装面からもアプローチすることができただろう。

あるいは、また別の機会に同じサイトを見に行き、 今度は 「Open-Closed Principle とデザインパターン」 を読んでみたとしよう。 ひょっとして「OOSC」こと B. メイヤーの「オブジェクト指向入門」を 読んでいれば、OCP（開放／閉鎖原則）くらい知っていたかもしれない。 けれど、あなたも私のようにその当時にはまだ読んでいなかったので

あれば、 まだまだ聞きなれない言葉だったはずである。 その OCP の紹介記事でもあり、また「デザインパターン」に関する ものめずらしい、しかしながら本質的な視点からの解説記事でもある この文章を読めば、 カタログとしてフラットに並べられたパターンに対して、 OCP という明快な切り口があることを、 平易な説明を通して理解することができただろう。

あるいは、あなたが 2003 年か 2004 年ごろ、この Rubyist Magazine が 発刊される以前に、Ruby の Win32OLE を使おうとしていたとしよう。 そのときにはもちろん arton さんの優れた Windows における Ruby の解説書、 『Ruby を 256 倍使う本邪道編』は出ていた。 が、ひょっとして ASR を使わないため読んでなかったとか、 あるいは自分のやりたいことが直接書かれていなかったりとか、 そもそも Windows の API やらオブジェクトモデルやらに明るくないとかで、 もっと他のサンプルを探していたかもしれない。 であれば、あなたはきっと検索エンジンか何かのお世話になり、 「Ruby による Win32OLE プログラミング」 のページにたどり着いていただろう。 今でもそうだが、Rubyist Magazine で掲載されている Win32OLE の記事もまだ存在していない 当時では、Win32OLE に関する実践的な解説として、とても貴重な文書だった。

あなたはきっと、このサイトにあるめぼしいコンテンツを一通り読んだ後、 この人の書いたものを他にも読みたい、と思うだろう。 さらには、この人に会ってみたい、機会があれば話をしてみたい、 と思うかもしれない。

けれど、もし今までそのような機会に恵まれていなかったのであれば、 それはもう叶わない。叶う日は、永遠に来ない。

> 「とてもはっきりしている処から、話しましょうか。……あのね、 人はね……死ぬのよ」
> (新井素子『チグリスとユーフラテス』(集英社) より)

人は死ぬ。いつかは死ぬ。だから問題は、いつ死ぬのか、である。

もう死んでいいという時があるとは思いたくない。 けれど、死ぬにはあまりに早すぎる、 と誰しもが思ってしまうことは確かにある。 そんな死に方をしていいのか、そんなときに亡くなってしまっていいのか、 と思ってしまうことはある。 そして、それなのに、そのようなことが起きてしまうことがある。 そのようなことが現実になってしまい、 あるべきことと起きてしまったことの落差に愕然としてしまうことがある。

訃報は唐突だった。どこで見たのが最初だったかは覚えていない。 けれども嫌な感じ

はあった。いくつかのサイトで気になることが書かれていた のを見て、ひょっとして、と思った。間違いないと知ったのは、 結城浩さんによる石井勝さんことまさーるさんの訃報のページだった。

家に帰り、まさーるさんのサイトに残された文章を読んでいるうちに、 思わず嗚咽が漏れた。 会ったことなど一度もないのに、 声を聞いたこともないのに、 顔を見たこともないのに、 もしかするとどこかのイベントですれ違っていたかもしれないが、 とにかく一面識もない方の訃報なのに、 涙が止まらなかった。

繰り返すが、私とまさーるさんの縁はほとんど何もない。 ネット上で、メールやその他の場所でのやりとりすらかわしたこともない。 私が勝手にまさーるさんの書かれた記事を読んでいただけだ。 ObjectClub - 石井勝さん追悼 のページに書かれているような、日本のオブジェクト指向界での「数多くの功績」のうちのごく一部しか知らない私は、 本当に純粋な、Web に掲載された文章の書き手と読み手の関係しか持ち得なかった。 そのような方への追悼の言葉を書き記しても、 どこかよそよそしいものにしかならないような気がする。

ただ、Ruby をもっとよく使いこなしたい、オブジェクト指向をもっとよく 理解したいと思っていた数年前の、そして現在の私にとって、 まさーるさんの残した文章がどれほどの影響を持っていたか、 私がどれほど助けられたか、 それだけは書いておきたかった。ドキュメンテーションなどで Ruby と Rubyist に影響を与えた方の中には、 Ruby のコミュニティ内ではあまり目立たれていない方も少なくないと思われる。 そのような方の一人として、まさーるさんという方がいたことを、 Rubyist Magazine を読まれている方々には、知っておいてほしかった。

まさーるさんが書かれた文書は、 確かに自分の血肉の一部になっていると思う。 そのような優れた文書を書かれたことに、 心から感謝している。 そして、それから得たものを糧にしつつ、 まさーるさんが書かれていたような、 オブジェクト指向やソフトウェア開発や Ruby について語る文章 （それがまさーるさんの書かれていたようなものに適うかどうかはともかく）を 掲載する場を提供している雑誌に名前を連ねていることを、 心から誇りに思っている。

Rubyist Magazine 第 6 号をお届けする。

Rubyist Magazine 巻頭言選集

2013 年 2 月 8 日 v0.9.0 版発行

著　者　高橋征義

発行所　達人出版会