Tim Maher

Is it Time for Perl Certification?

This month, Tim Maher calls for the Perl community to adopt a serious certification program, both to help individual Perl programmers make it through the hiring process, and to increase Perl's standing in the corporate IT market. What do you think? Your comments are welcome at editors@tpj.com.

— Editors

During my six years as a Perl educator and contract programmer, I've often pondered the problems facing the Perl community. One of these is the alarming state of un- or under-employment of many Perl specialists (including some of our brightest stars), while our colleagues who program in more prosaic languages such as Java and C++ enjoy more secure job positions.

To gain a better understanding of how Perl and its programmers are perceived in the industry, I've talked with managers of Information Technology (IT) departments and professionals working in Human Resources (HR) and recruiting agencies. This has led me to some conclusions about how we could improve our situation that I'm eager to share with you.

But first, you should know some of the pertinent aspects of my background. During my 12 years in academia, I obtained a wealth of experience in both taking and constructing examinations, and I studied techniques for computer assisted learning and testing. Later, while working with Sun Microsystems Inc., I had to take and pass the certification exams of the "Solaris System Administrator" series, and I provided feedback to help improve their quality.

These experiences have made me comfortable with testing technologies, but also highly cognizant of the need for testing to be done accurately and responsibly.

Before I turn to the subject of Perl certification, I'll review certain aspects of Perl's current status in the enterprise.

Perl's Image Is Cool, but Strange

It goes without saying that Perl is a marvelously expressive, extensible, and productive language that's fun to use. This has been noticed by many IT departments that value it as a general-purpose scripting language, a language for CGI or DB development, or one for cross-platform system administration.

Unfortunately, from the vantage point of old-school computer science types, Perl can also look like a "toy language" when compared to ones such as C++ and Java. That's partly due to Perl's lack of support for many features that let IT managers sleep better at night, such as strict type checking, compile-time function binding, standardized exception handling, and a conventional OO model.

In consequence, when it comes to critical software development projects — you know, the kinds that retain their participants even during recessions — Perl doesn't make the grade. It's summarily rejected for such projects, because their participants tend to view characteristics such as successful compilations in the face of missing subroutine arguments as *tragic flaws*, rather than charming indulgences of poetic license.

But Perl's historical laxity in these areas is by design, because Whipuptitude and Manipulexity (Larryisms, of course) are largely incompatible with, and have always had higher priority than, those aforementioned sleep-inducing properties.

Perl Should Cater More to Corporate Needs

Besides the intrinsic strengths and weaknesses unique to Perl's design tradeoffs, I believe there's another reason why Perl is often left sitting on the bench in the big games. Put bluntly, we've done a lousy job of pitching our language to the business community.

For instance, there are many IT managers who are willing to consider Open Source solutions, but are unsure how to view Perl relative to the languages they know. And that's perfectly understandable, because Perl's eclecticism is associated with lots of mixed messages, that would confuse any sensible person.

They're told that Perl is a scripting language, but not really, because it's compiled, but then again it's some strange kind of compilation that doesn't produce object code, so it's not really compiled, and it's a procedural language, or sometimes an OO one, but there aren't really classes or exceptions, and there isn't just one OO model—you can roll your own, and it's a *feature* that missing functions won't trigger warnings until they're called, rather than at compile time, and so forth.

Tim is the founder and leader of SPUG (Seattle.pm) and the CEO of Consultix, which offers Perl, UNIX, and Linux training. He can be reached at tim@teachmeperl.com.

Not to mention the fact that Perl programmers are notorious for cultivating idiosyncratic dialects of the language, that are almost as distinctive as handwriting styles, and can prevent one programmer from being able to read or maintain another's programs.

On top of all this, Perl advocates are inclined toward pronouncements like "It's *too hard to parse* to allow beautification," "It's so flexible it can be programmed in *poetry mode* with *autovivification* and *bleached code* or in *Latin* with *Klingon numbers*," and "It's too *multifaceted*, *expressive*, *advanced*, and, well, *artsy* and *esoteric*" to allow meaningful certification tests for its programmers.

(Regarding beautification, see my 1998 TPC presentation on the first "Perl Beautifier" at http://www.teachmeperl.com/perl_beautifier.html.)

Furthermore, as the last nail in the coffin, the only "Perl certification" identifiable with the Perl community itself is a *purposely bogus one*.

Given the technical shortcomings just summarized, and PR like this, is it any wonder that other languages, including the most stodgy, cumbersome, and uninspired ones, are eating Perl's lunch in thousands of corporate cafeterias?

But there's a development in the offing that could help improve our situation.

Perl 6 Will Have More Enterprise Appeal

The wondrously reworked Perl 6, when it arrives, will go a long way toward stemming many of these concerns about the fickleness and eccentricity of our language, because it promises to give programmers the option for more rigor right where the IT managers want it. That means they can have the best of both programming models—the traditional "expressive and intuitive and loose" Perl, and a new, "more rigid and bulletproof" variation. It will still be Perlish, but suddenly no longer a weak sister to C++ or Java.

Perl 6 is justifiably expected to stimulate a widespread reconsideration of the important roles that Perl can play in IT departments, and an increase in new hiring for JAPHs (JAPH means "Just Another Perl Hacker").

But how will hiring managers be able to confirm that the applicants for the new Perl jobs really know Perl 6? An answer that would readily occur to those managers would be "through *certification.*"

But while we're waiting for Perl 6, is there anything we could do to improve Perl's image?

How Can We Help Perl Get the Respect it Deserves?

What would be the best way to improve the (somewhat motley) position of Perl in the corporate world, so it would be more frequently chosen for important applications, and Perl programmers could get placed in more secure positions?

How about improving Perl's documentation? That couldn't help; Perl's documentation is already the best in the industry.

How about creating a worldwide network of support groups for Perl programmers? We've already got one. It's called "Perl Mongers."

How about making Perl more robust? Perl 6, which will be more robust in all the right areas, is on the way.

How about launching a PR campaign extolling Perl's virtues? It's probably too late for Perl 5, whose strengths and weaknesses are already well known, for any attempt at "spin" to be very successful. We'll have a lot more to boast about when Perl 6 comes out, so it might be best to delay this kind of effort until then.

How about creating a certification program for Perl programmers? Hmm…that might be just what we need!

Establishing Perl skills as *certifiable*, and the Perl community as willing to comply with accepted hiring protocols, could cast Perl in a totally new light. First of all, hiring managers would be inclined to see Perl as more stable and conventional, because certification (unlike "poetry mode" and "bleached code") is considered a hallmark of serious languages. Second, they'd realize that screening Perl programmers would suddenly be no more difficult than screening Java programmers, Oracle Database Administrators, or Linux System Administrators.

If we don't rise to the challenge to do it properly, some opportunistic corporation might beat us to the punch

And of course, these benefits at the hiring end of the equation would make it easier for managers to consider basing additional projects on Perl, because of the greater ease in staffing them.

Finally, when Perl 6 arrives, managers would realize that the new Perl would be suitable for the most critical enterprise applications, offering additional incentives for increasing Perl development and JAPH hiring.

In a nutshell, these are the conclusions I've come to, and my recommendations to the Perl community. In the remainder of this article, I'll provide some details that will help you see how I've arrived at these conclusions, and help you make up your own mind about this important issue.

There's a Demand for Certification

As a quick search with Google will confirm, there are several vendors currently offering certificates of Perl competency based on online tests, and one of these has reportedly been designated as a requirement for job applicants at certain companies.

The very fact that some have found it necessary to qualify applicants for Perl jobs on the basis of certificates of dubious value (more on this in a moment) indicates a real need for a legitimate Perl certification service that's going unfilled.

Through my involvement with the Perl community, I've made contact with many hiring managers who are also Perl programmers and advocates, and they tell a similar story, which goes like this: They'd like to use Perl more widely in their (old-school, traditional) businesses, but they feel like they're swimming against the corporate tide. Eventually, they get tired of championing an underdog, and they ultimately settle on another language that's easier to defend to colleagues and more amenable to HR screening practices. (This reminds me of the old "Nobody ever got fired for buying IBM" ads that appealed to the CYA demon whispering in every manager's ear.)

One of the major complaints of these managers is the lack of any help from the Perl community in validating the skills of a job applicant. That puts the burden of vetting essential applicant skills directly on the shoulders of the manager (or his staff). And they know that's not a burden shared by managers hiring Java programmers, because the Java language is associated with a series of professionally designed and administered certification tests which are widely respected as evidence of competence.

But what about C++, Perl's other major competitor in the enterprise? Like Perl, it lacks a widely accepted certification

program, but that certainly hasn't prevented it from reaching widespread acceptance.

The crucial difference between Java and C++ is that C++ was there first, and had an existing base of corporate interviewing teams that felt comfortable hiring C++ programmers. When Java emerged as an alternative, its proponents had to work harder to make it appealing to the business community. That's why they went to the effort of developing a certification program, to make it easy for HR departments to perform initial screenings of Java job applicants, and to make the new language easier for managers to adopt.

We in the Perl community are in many ways in the same "underdog position" as Java initially was, so we'd be wise to take a page from Java's book, and make it as easy as possible for companies to hire Perl programmers.

Clearing the First Hurdle

Larry Wall's whimsical thoughts on the "Three Virtues of Perl Programmers" are well known in the Perl community. An even more essential bit of knowledge for JAPHs is the first hurdle that must generally be overcome by a job applicant, especially when dealing with a big company. That hurdle, called "screening," is erected by the HR department, and those who surmount it get their résumés onto the hiring manager's desk, while the others have theirs consigned forever to the dreaded *circular file*. (The same process is also commonly used by recruiters working for placement agencies, but for simplicity, I'll refer to this as an HR activity.)

The important thing to understand about HR departments is that their technical knowledge is limited to buzzwords and certifications. Accordingly, when 200 résumés show up on Monday for the single Perl job advertised on Sunday (as can happen in the USA), HR starts the screening process in a frenzy, to reduce that stack to the much shorter one desired by the hiring manager.

During the winnowing process, in the absence of explicit instructions to the contrary, résumés that are missing that prized screening credential—a relevant certification—generally get trashed. In fact, the screeners might in their zeal even trash the résumé that says "Larry Wall" at the top, if they're lucky enough to get it.

In one case reported to me, this ruthless screening process caused 100 percent of the applicants for a Perl-only position to be ruled ineligible for an interview! In other situations, such as positions that invite both Perl and Java applicants, this process has put JAPH contenders at a huge disadvantage to their (more commonly certified) Java competitors.

OSCON Attendees Voted *For* Perl Certification

At the 2003 O'Reilly Open Source Convention (OSCON), I moderated a Panel Discussion on Perl certification (http://conferences.oreillynet.com/cs/os2003/view/e_sess/3747), featuring a diverse and distinguished panel, including Perl 6 designer Damian Conway. The discussion centered around the pros and cons of having a certification program for Perl programmers.

Approximately 200 people attended the session, and during the open discussion period, many posed questions to the panelists, and shared their own experiences and views. As any seasoned observer of the Perl community would guess, passionate arguments were heard on both sides of the issue.

But something happened at the end of the session that surprised all the panelists, and every Perl community "leader" with whom I've since discussed it (but interestingly, not many Perl "followers"). Specifically, Damian called for a show-of-hands vote of attendees *for* and *against* the development of a certification program for Perl programmers, and the *for*s won by a wide margin of approximately 14 to 1 (as agreed by the author, Damian, Nat Torkington, Tim Wilde, and others; for further details see http://teachmeperl.com/perlcert/OSCON_vote.html).

That outcome was a surprise because previous discussions on this subject had not shown a strong majority in favor of certification. But the fact that those prior exchanges were often dominated by influential figures arguing against the idea provides a possible explanation for this discrepancy. Specifically, I suspect that the less-famous members of our community might have been reluctant to go on record in a public forum as expressing disagreement with the more famous ones.

But unlike the environment offered by a newsgroup, a mailing list, a wiki site, or a spirited discussion in a pub, the OSCON show-of-hands vote should have provided an environment where individuals could feel more free to express their views on this controversial topic—after all, their names were not even requested, let alone stored on the Internet for all to see.

We've done a lousy job of pitching our language to the business community

This vote provides the most specific and credible evidence we've ever had of community opinions on this topic, and it tells us that JAPHs are clearly in favor of making certification a reality.

Certifications

What exactly is certification? Professionals, in a wide variety of specialized fields, obtain certification as a way of establishing their knowledge, whether to satisfy licensing conditions imposed by regulatory authorities (CPAs, attorneys, doctors, auto mechanics, and so forth), or as an aid in convincing prospective employers of their skills.

Many software technologies have serious, standardized certification programs including Java, VisualBasic, Visual C++, Oracle, DB2, and various flavors of UNIX and GNU/Linux.

Historically, Perl programmers have had the opportunity to acquire four types of credentials attesting to their knowledge of Perl. These credentials vary widely in price, sophistication, credibility, and value, and they are listed below.

School certificates: In recent years, certain progressive institutions of higher education have established Perl training programs that award certificates to their graduates. I'm on the advisory board for such a program at the University of Washington.

That program covers a fairly comprehensive range of programming topics, and awards a pass/fail grade based partly on the student's ability to submit acceptable source code. Instructors have included Perl specialists from Amazon.com and the Slash project, among others, who are familiar with current Perl programming practices in both corporate and Open-Source development environments.

Because students who earn these types of certificates have not only learned practical uses of the language but have also demonstrated an ability to program in it, these are generally considered the most impressive certificates currently available.

Training vendor certificates: Many training vendors award "class completion certificates" to students who attend training classes. Although there was a distant time when it was not unusual for final exams to be administered on the last day of such classes, in recent times, the industry standard has shifted toward

awarding these certificates largely on the basis of *attendance* (can you say "self esteem movement"?).

Naturally, such credentials are of rather limited value in themselves, but they still signify something valuable. That's because the vast majority of students attending such classes actually apply themselves and learn the material.

Dubious certificates: With a little Googling, one can find a few vendors that are currently offering Perl certifications, based exclusively on brief on-line tests, at very low prices ($50 or less).

But even if these tests were masterfully designed and initially validated under controlled conditions by accepted psychometric procedures, the certificates they're now awarding would have to be viewed with circumspection. That's because these vendors are testing individuals with unconfirmed identities under unregulated conditions, which allows:

- Anybody to pose as John Doe to get "him" certified.
- The real John Doe to take the test, while obtaining answers from somebody else.
- The possibility that a prospective future testee could print the test, get somebody to tell him the answers, and then later pass the test—without ever learning anything about Perl! (This assumes the test questions are drawn from a small pool, which is a safe bet for such cheap programs.)

These uncontrolled testing conditions represent an egregious violation of psychometric requirements, including "validity," the property that the test (as administered) is truly assessing the testee's knowledge of the subject, and "reliability," the property that a similar grade would be expected on a retest.

Joke certificates: There are well-known members of the Perl community who are strongly opposed to certification. In fact, in an exhibition of admirable entrepreneurial spirit, a few of them started selling fancy and personalized, but *totally bogus* Perl certifications years ago. Their admitted motivation is to make it difficult for any serious attempt at a Perl-certification program to gain acceptance, by flooding the market with these cheap, fraudulent certificates, which hiring managers will think are legitimate.

It's understandable that anyone who had been offended by a laughingly defective certification test would find the cynicism underlying this prank to be amusing. But despite its Pythonesque appeal, this "disservice" has, thankfully, not caught on.

But I shudder to contemplate the message this is sending about Perl to the IT community, which is one of disrespecting accepted corporate hiring practices, and actively plotting to preserve the current difficulties that HR professionals face in hiring JAPHs.

Serious Perl Certificates

Never in Perl's history has there been a Perl certification program that was widely recognized by employers or endorsed by community leaders. Far from it. I think we in the Perl community should change this, by creating a *serious* certification program.

Its tests should be:

- Designed by subject matter experts.
- Compliant with established psychometric principles (validity, reliability, and so on).
- Administered under regulated conditions.
- Controlled by a respected organization.
- Endorsed by community leaders and leading corporations.
- Sensitive to feedback from testees, to facilitate identification and correction of (inevitable) errors.
- Optional, based on an understanding that one's experience and track record should be recognized as alternative ways of establishing one's qualifications.

Beneficiaries

The beneficiaries of a serious certification program for Perl programmers would include:

- Recruiters, HR departments, and hiring managers who need assistance in screening, classifying, and ranking applicants for Perl jobs.

Establishing Perl skills as certifiable could cast Perl in a totally new light

- Perl programmers, who would have an opportunity to obtain Perl credentials that would be compatible with established corporate hiring practices, and who would have help in identifying the gaps in their knowledge, so they could better their skills.
- Booksellers, publishers, authors, training vendors, colleges, and testing centers, because there would be an increase in demand for Perl-related educational materials and services. (See http://teachmeperl.com/perlcert/beneficiaries.html).
- The whole world of Perl, through a formal definition of the essential components of the basic language and the specialty areas; through the enhanced professionalism of our image that would more accurately reflect the value of our language to the enterprise; and through revenues flowing back into the community, if certification becomes profitable.

Perl Certification as a Rorschach Test

The phrase Perl Certification seems to conjure up a nightmare version of a Rorschach Test for some JAPHs, who imagine the worst possible interpretation of an ambiguous stimulus, and react accordingly. They picture an obligatory, monolithic, multiple-guess trivia test that strives to encompass everything Perlish, takes five hours, and is conspicuously missing correct answers for 100 of its 500 questions.

What's worse, it has to be taken at a special test center, in the next major city. And it must be retaken annually. Oh, and it costs $500 each time. Not including tax. But no checks or charge cards are accepted, only U.S. dollars. Small bills only, $20 max. Must be in mint condition…

A test like this, which is not (altogether) unprecedented in the industry, would function as more of a "programmer tax" and "anger management challenge" than a test of Perl knowledge. But, this nightmare scenario aside, I suspect that most of certification's detractors wouldn't really oppose a well-designed, well-executed, sensible, and fair certification program, especially if it could help our community, and if they had direct input into its design.

Why Rock Perl's Boat?

Okay, so Perl has survived all these years, acquired a large following, made some inroads into IT circles, and accomplished all that without the benefit of a respectable certification program. Why should we make such a dramatic change as to adopt a certification program now?

I'll give you four reasons:

Perl skills are not properly valued. In my capacity as the job-listings liaison for SPUG (aka Seattle.pm), I notice changes in the job market for Perl programmers, and it's obvious that they have suffered disproportionately during this extended recession.

One disturbing trend is that many IT managers have retained their Java and C++ programmers in recent years, while laying off their Perl programmers. Of course, some of this layoff disparity is rightfully due to the "mission critical" nature of the jobs some Java and C++ programmers are doing, versus the less glamorous "glue" jobs performed by many Perl programmers. (But if those managers only knew how much of the useful output from their Java and C++ programmers was really derived from clandestine Perl usage, they might have more respect for their JAPHs!)

These JAPH-dumping IT managers are probably the same ones who practice lunchtime economizing by skipping the previously routine $3.45 latte, while maintaining the tradition of the Whopper or Big Mac. And of course, Perl (and its JAPHs) would be better off in the entrée category!

I've also noticed another aspect of JAPH under-appreciation. In recent years, many smart and capable members of SPUG, after being laid off from their Perl jobs, have found it necessary to earn certifications in Java or C++ in order to gain new employment.

Wouldn't it be better for the Perl community if they could secure employment by obtaining certification in Perl *instead*? With the arrival of the newly robustified Perl 6 that will make Perl a favorable alternative to its competitors, and a "business makeover" to allow Perl's PR to more accurately reflect its newfound capabilities, I think we could change the workplace into one where that could happen.

Perl 6 heralds a clarion call for certification. In my optimistic vision of the future, I see a world where the Perl 6 team members will all have sufficient job security that they can devote more time to their back-burner activities, and finish Perl 6. And it will garner great reviews, and rekindle the interest of IT executives, who will be encouraged by its obvious superiority to give the new Perl a chance for more widespread use in the enterprise.

By then, the U.S. economy will have had plenty of time to turn around, so we might even experience a new Golden Era (or more likely a Bronze Era, but that's good enough) of high-tech corporate hiring. You know, sort of like 1998–2000, but without the unsustainable and unhealthy exponential components.

And guess what the first question will be that recruiters, HR executives, and hiring managers will be asking applicants for the scores of newly allocated Perl 6 development positions? I'll tell you: "Before we go any further, exactly how much do you know about Perl 6?"

The advent of Perl 6, with its greater rigor, robustness, and "conventionality" in certain critical areas could be our golden (okay, *bronze*) opportunity to win market share from other languages. But how easily that will be accomplished will depend on how easy we make it for corporations to come up with confident assessments of our Perl 6 skills, to hire us, and then to be dazzled by the wonders we can achieve with it.

But to make the most of the opportunity provided by Perl 6, we should have a mature and respected certification program already in place when it arrives.

Otherwise, no matter how interested IT managers might be in giving Perl 6 a chance, there will still be significant corporate obstacles blocking its widespread acceptance—such as HR departments that don't know what to do with résumés from JAPHs except to file them in the wastebasket.

The boat is already rocking. The pertinent question is not "Why rock Perl's boat," because in fact it's already rocking. The more appropriate question is "How do we keep it from capsizing?" We're losing market share to other languages, such as Ruby, Python, and PHP, whose main contribution is providing more prosaic and conventional ways to get at Perl-like capabilities.

I've even heard of IT departments hiring nonprogrammers and teaching them to write CGI programs in PHP rather than hiring experienced JAPHs to do those same jobs with Perl, because they don't feel comfortable with Perl's unrivaled peculiarities, complexities, and freedom for individual expression. We need to work on reversing such trends in the marketplace, and soon.

Perl 6 could be our golden opportunity to win market share from other languages

Certification is worth a try. I know of no case where the introduction of a serious and well-managed certification program for a software technology has ever resulted in changes that were largely detrimental to the associated community. Sure, there tends to be some initial grumbling about the testing fees by those who choose to seek certification, but if it helps them obtain and keep employment, and improves the image of their technology, and increases the value attached to their skills, that's all beneficial.

We know we have problems competing for enterprise programming jobs now, and that the advent of Perl 6 will create new demands for certification. Sitting on our laurels surely won't help us face these challenges, but developing a certification program seems likely to help. There are risks involved (I'll get to these in a moment), but I think the odds are in our favor—in large part, because we, in the Perl community, as the developers of the program, would be in charge!

That means we'd have the freedom to devise any kind of unconventional certification regimen we might fancy—so long as its results are reducible to a few words on a résumé, for the convenience of HR departments. For instance, in keeping with the TMTOWTDI principle, credit could be given for knowing *any* correct solution to a programming problem, rather than a particular one, so those who know different dialects of Perl could all obtain certification.

And the "HR-friendly" words associated with testing might include "Perl Certification, Level 1: Passed with Distinction," as well as "Acknowledged Perl Guru" (for those granted testing waivers, on the basis of their code portfolios)—if that's what we want.

The Downside

I've outlined my views about how a serious Perl certification program developed by our community could benefit us. But as any maintenance programmer can tell you, the introduction of any new element into a complex system raises the probability that things will go awry.

My perspective on these concerns is simply this: If we take on the responsibility for this task as a community, we'll be in charge. So if we see things going awry, we can take corrective action. If we do this well, individual JAPHs with good ideas will have a much better chance of effecting changes in Perl certification than they would have with other programs, run by large corporations, that are looking out for their own vested interests.

Certification only needs to yield a net gain to be a success. There will undoubtedly be some undesirable repercussions of

introducing such a program so late in the evolution of Perl and its community. But if the results are largely beneficial, especially in the critical employment arena, we'd be foolish not to seize the opportunity to make the world a better place for JAPHs.

And we must not overlook the fact that there are also risks associated with inactivity. Specifically, if we don't start developing our own serious program for Perl certification, and soon, somebody else might do it for us—or perhaps the more appropriate phrasing would be *to us*.

Policy Precedes Implementation

Although I've written and spoken elsewhere about my own ideas for implementing an optional, state-of-the-art multilevel certification program for Perl, and several others have also offered creative ideas, I've purposely avoided matters of implementation here.

That's because the unavoidable nitpicking involved in design and implementation must not be allowed to get in the way of rational decision making, and we have a very important decision to make at this juncture.

Given that a majority of community members expressed a desire at OSCON for Perl to have a certification program, the question we must now collectively answer is: "How should we respond to the demand for Perl Certification?"

If we decide to "make it so," then we should proceed to collectively determine what the properties of that program should be and work on creating it. But there is one formidable obstacle that we'll have to overcome.

The Biggest Obstacle: Us!

We in the Perl community are both our greatest asset and our greatest liability (but life's like that!). If an upstart language as great as Perl had been created by a corporation (think Java, but with more inspiration), the business-friendly infrastructure would have been incorporated from the start, just as surely as The Larry felt the need to provide **a2p** and **s2p** with the first release of Perl. So that hypothetical language would have had an effective PR program and a certification process long ago, and the advantages that accrue from them. (By the way, **a2p** and **s2p** are, respectively, awk-to-Perl and sed-to-Perl translators. Larry provided these to automate conversion of programs written for those UNIX utilities into Perl programs, with Perl's initial release. What a guy!)

However, along with having greater intelligence, creativity, generosity, sociability, and tribal spirit than your average geek, JAPHs also exhibit greater independence and nonconformance. And that (sometimes unfortunately) includes a tendency to flout the established conventions of the corporate world.

But this is to be expected—it's no accident, after all, that we've been brought together under the banner of TMTOWTDI. Nor is it an accident that most of the greatest accomplishments in Perl culture (CPAN, CPANPLUS, DBI, PAR, perltidy, the Perl 6 design, the Parrot interpreter, TPJ, the Camel book, the Perl Cookbook, Damian's OO Perl book, TPF development grants, Perl Mongers, Perlmonks, and YAPC) have been achieved either by individuals working alone, or in very small groups of like-minded colleagues.

In recognition of this, it would seem that our best chance for success would be to have a small group of people, who show evidence of being able to agree on certain core principles, oversee this project.

But we'll have to avoid getting dispirited by those who will warn us of "insurmountable problems" and a "pestilence on all of Perlity" if we dare to establish such a program. Although this kind of input can sometimes freeze people into inaction, we need to remember two things: 1) Many groups have already done what we'd be doing (for example, the Linux community), and 2) We have an unmatched pool of talent, ingenuity, and perspicacity in our community to apply to the effort.

So there should be absolutely no doubt about our ability to make this happen, and make a success of it, if we should choose to take on this task.

Conclusion

Why should we create a certification program?

- HR departments want JAPHs to be certified to facilitate preliminary screening of job applicants.
- Managers want JAPHs to be certified to make their individual skills easier to compare.
- Catering to accepted hiring practices should promote greater hiring of JAPHs and greater acceptance of Perl in the enterprise.
- The advent of Perl 6 should cause renewed interest in Perl, more Perl positions in mission-critical application areas, and therefore, a heightened demand for applicants certified as having the latest Perl skills.
- Invalid "certification programs" that tarnish Perl's image should rightly have to compete with a serious one created by those who really know the language and have its best interests at heart—the Perl community.
- Vendors are already providing this service with various degrees of sincerity and sophistication. If we don't rise to the challenge to do it properly, some opportunistic corporation might beat us to the punch, and wrest control of this important aspect of our culture from us.

It's Time for Perl Certification

My recommendation is that we immediately start working on a certification program for the essential skills of Perl 5. Once that proven testing infrastructure is in place and accepted by the corporate world, we can start creating new tests for Perl 6, while continuing to develop add-on certifications for Perl 5 (which should remain an important language through the rest of the decade).

How to Get Involved

It would seem most natural to develop a community-based Perl certification program under the auspices of a community-wide organization. Although The Perl Foundation is the only one we have that comes close to being appropriate, and its leader is interested in this issue, she has thus far declined to take an active role in it.

But if we're to make any headway on this challenge, somebody will have to act as a coordinator—so I volunteer.

As a first step, I encourage those interested in helping to develop a Perl certification program to subscribe to the Perl Certification Mailing List, at http://perlocity.org/cgi-bin/mailman/listinfo/perlcert. Then post a message announcing any special qualifications you might have (such as experience in academic testing or corporate hiring) and indicating what roles you might be willing to play in this effort (test designer, fund-raiser, business liaison, hosting provider, and so forth).

We'll take it from there—as a community!

Acknowledgment

The author is indebted to the following reviewers for helpful comments on earlier versions of this article: Damian Conway, Terry Nightingale, Rodney Doe, Nancy Corbett, Christie Robertson, and John Michael Mars.

TPJ