Integrating Perl in a wider distribution: The Debian pkg-perl group

Gunnar Wolf
Instituto de Investigaciones Económicas — UNAM
Debian Project Developer

August 21, 2007

Abstract

Perl modules are very well organized in CPAN: They can usually be easily found and, thanks to tools such as the CPAN shell, they are easy to install and update even by novice users. However, when people start using Perl systems (as opposed to using Perl for *writing* such systems), asking them to take care of the dependencies or having them worry about different distribution architectures is a pain that should be spared from them.

In my talk, I will describe how Debian [Debian] (and other Free Software distributions) addresses this problem by packaging a large subset of the CPAN archive, what is the task and scope of Debian pkg-perl [pkg-perl] team, some of the tools we use - and, most importantly, what is the best way for us to interact with you, the upstream authors' community — regarding our bug tracking systems, regarding module building and dependencies information, etc.

1 Who and why?

1.1 Who am I?

This article was originally written by Gunnar Wolf <gwolf@debian.org> to be presented at YAPC::EU 2007 [YAPC::EU], with the spirit of being able to spark interest and further comments both from the Perl *and* the Debian communities, hoping to interact better and provide a simpler and more consistent experience for our users, at all experience levels.

I have been a Perl programmer for almost ten years already, and although I am far from considering myself an expert, I have presented talks both in YAPC::NA 2001 and YAPC::EU 2002. I maintain two (small, simple and low maintenance) CPAN modules, and although I have been lured towards differnt languages and ways of working, my *mother tongue* as far as programming languages are concerned is still (and will probably always be) Perl.

By 2001 I also started getting involved in Debian, and since 2003 I am officially a Debian Developer as well. My main work area in Debian is in the pkg-perl group, and I have recently started getting involved with the pkg-ruby-extras group, which has similar goals, although it is targetted at the Ruby language.

1.1.1 Why am I presenting this? And why here?

Every programmer wants their work to be useful — Or at the very least, to be *used*. In the Free Software world, one of Perl's most successful *ecological niches*, the best way for our work to reach the users is to get integrated in a major *distribution* (Refer to section 3.2 for a more complete explanation on the distributions, their goals and ways to achieve them)

To be honest, I expect giving this talk will almost be like *preaching to the choir* — At Debian, we have almost had no problems working closely together with the Perl developer community. Some of the tools that sparked the pkg-perl's efforts were even based on work by prominent members of the Perl community [Brocard]. I must thank you all: The Perl community is a charm to work with. I think the main reason for it is that it is a mature community, that mostly clearly evolved from Unix users and administrators — people clearly compatible with Debian's way of working.

My aim with this talk is to present to the Perl community the work we do at Debian, in order to get input on how to better work together, and of course, to share experiences and points of view with other distributions' packaging teams.

1.1.2 Are we really that similar?

There are some other coincidences to this, although I don't know if there is any deeper sociological reason for both communities to work in such a similar fashion.

First, Perl modules are often excellently documented. Although documenting is —fortunately— often seen as a very important value among Free Software developers, it is often left to the end of the development cycle, or covered hastily just so nobody complains. Debian's policies, however, state that the lack of documentation is considered a severe bug, enough to block a package from entering its stable releases. With Perl modules, this is seldom a problem — Contrary to some other language communities (think Ruby, PHP) which stress more *the agile way*, where documenting falls out of favor because it encourages not spending so much time in the design phase, as posterior iterations will probably change the base behavior of the system. In CPAN, even the least maintained modules have basic documentation available.

Second, the Perl community has gone past the rapid development stage where the community requires every component to be up to the latest minor version. The Perl community values dearly supporting older versions, not hastily deprecating functionality as soon as new ways are developed. Perl, just as Debian, has been blamed for stagnating — A claim often made, of course, by people who don't see the technical excellence behind, and who don't value long-term stability and predictability.

Many of the thousands of modules available in CPAN 2.1 have stabilized in providing the functionality they intended, and can often spend many years without being modified — This does not mean they are abandoned (bugs are usually dealt with promptly by the authors), but that they are stable and solid.

2 CPAN background

One of Perl's longest standing *selling points* is CPAN. It is probably the largest repository of this kind in the Free Software world, and an immense source of resources for its programmers.

2.1 In the beginning, there was CPAN, and it was good

The Comprehensive Perl Archive Network [CPAN] is the prime resource for a Perl programmer. It has been online since October 1995, and as of May 2007, it hosts 11577 modules from 5856 distinct authors [CPAN Modules, CPAN Authors]. Not even that, it is also a fast-growing and fast-evolving list: During 10 days in May, the total number of modules grew by 76. Every day, there are over 20 new uploads [CPAN Recent].

CPAN is, in no small part, responsible for Perl's success: Besides being a very enjoyable and powerful language to develop with, but it has a tremendous number of proven, categorized modules to ease an individual programmer's task.

An important feature of CPAN is that it is a centralized repository, which allows us to treat it as a single entity and perform operations throughout the module base. A homogenous quality among CPAN modules is far from reality, but there *is* a base number of features CPAN's modules have[1] — Standardized module layout, common build systems, decent test coverage, a centralized bug tracking system (to which the module developers *do* tend to pay attention)... In short, the Perl community is probably the easiest to integrate with Debian, as explained here and in section 1.1.2, both technically and socially.

[1]Feel free to just prepend an "almost" to the beginning of each assertion I make

2.2 And then, there is CPAN.pm

Although originally just the fact of having a centralized repository was good enough, very soon modules started becoming complex and depending on each other (even worse: On *specific versions* of each other) — After all, that's what code reuse is all about, right? Well, yes, but life for CPAN's users started deteriorating — Installing a module no longer was matter of downloading a tarball and going through the proven `perl Makefile.PL && make && make test && make install` cycle. And as CPAN started growing, it would only get worse.

It took very little to reach this point. By February 1996, Andreas König had realized this could only be solved by providing an infrastructure for modules to declare their cross-dependencies — and, more important, for a client program to solve them on behalf of the user. Thus CPAN.pm [CPAN.pm] was born. This module provides the user with an infrastructure for installing and keeping up to date all the modules registered in CPAN. For practical purposes, this solves the problem.

In 2001, Jos Boumans started working on CPANPLUS (also known as CPAN++) [CPANPLUS], a redesign on CPAN which aimed at modularizing all of CPAN's processes so it could be not only used as a tool, but as a library. For some time, CPANPLUS was seen as a natural replacement for CPAN.pm when it became ready — However, by now most of this work has been integrated into CPAN.pm, which is much more heavily rooted in the Perl community. Anyway, we will not go deeper into it. For this article's purpose, it's enough to state that both systems are an adequate answer for the *Perl*'s *community* needs on managing installation, updates and dependency information between modules.

3 Integration with distributions

Although Perl is a highly portable beast, running in every kind of environment, it is most pervasive in the Free Software distributions — specially in the Linux-based ones[2].

3.1 The bigger picture: End users' experiences

The solution presented in the previous section is, however, a bit lacking in the end. It solves the problem perfectly for Perl's most intensive users, the programmers who often install all kinds of modules, finding the best ones to suit their needs for each problem at hand, and —of course— those making up the target audience of this article.

End users are, however, a completely different story.

We often develop complex Perl systems to be installed by people who don't really have knowledge of the Perl community's tools and processes. Ideally, our systems' users will not even care what language the system was developed in, and will never get familiar with its carefully crafted tools. They are just interested in the implemented functionality. And the same story goes for other laguages — I could have written section 2.2 referring to PHP's PEAR or Ruby's Gems [PEAR, Gems]. Python has the Python Cheese Shop [PCS], which is only the repository, but does not include an infrastructure for installing, updating and dependency handling.

This seems to be good for Perl, PHP and Ruby — And it is a great value for their respective *developer communities*. However, it can become a nightmare for the end users. As I have just stated, our systems' users *should not even care what language a system was developed in* —- Even if many Perl people feel that having the user call CPAN.pm as a clever advocacy trick, language advocacy should not be targetted at end users [Cross]. Leaving aside whether language advocacy should be carried out in non-programmer circles, the problem is having each language support its own packaging infrastructure and idiosincracy *is just too complicated* for most end users.

[2]As of today, besides Linux, the most widespread Free Software-based operating systems are FreeBSD, OpenBSD, NetBSD, Sun's OpenSolaris and Apple's Darwin (the lower half of MacOS X). All of them include Perl as part of their available packages, and most of them do so as part of their core installation.

To put this in other words: Our job at Debian is to make installing whatever Free Software a user wants as simple as just typing `aptitude install package_name`[3] — All dependencies (and, as far as we can provide, all of the installation instructions) should be immetiately and automatically taken care of.

3.2 Free Software distributions: Different support tiers

One of the main reasons for Free Software distributions to exist is to provide users (and specially end users) with a coherent, unified system, easier to keep track of, to integrate and to keep properly working. Debian is not only the largest distribution, including over 15,000 source packages, but quite probably is also the one with the most strict policies regarding quality assurance and software freedom. While the project I work for is Debian, most of what will be discussed in this section applies to all other Free Software distributions as well.

When something goes wrong, end users will usually file a bug report through the Debian BTS, to their *package maintainer*[4] in Debian. Sometimes the bugs are filed against the wrong package, or a particular user's request is not precisely a bug report but a wishlist request, or even a misunderstanding on how the package is supposed to be used — The package maintainer should take care of finding which package it actually refers to. The bug might be caused or triggered by packaging mistakes or conflicts, so it should be fixed without bothering the *upstream developer*[5]. The package maintainer should only push the report upstream (and, of course, contribute to finding its solution) if it applies to the original developers' code.

The distributions' structure acts, then, as a first safety net, trying not only to make life easier and more manageable for its users, but also to help the upstream developers not be as busy as they otherwise would if they had to deal directly with users' requests.

[3]Or the point-and-click equivalent for the GUI-minded people (which means, most of the end users, to be honest) via tools like Synaptic

[4]The person or team that take responsability for a given piece of code. Package maintainers' job is to ensure the program is well integrated with the rest of the packages in the distribution, up to date and, in general, to make sure that the package complies with Debian's policy.

[5]As you might imagine, *upstream developer* is the term we use to designate the real authors of a package we maintain.

3.2.1 The other way around: Perl supporting the Linux distributions

The reverse workflow might be also interesting for many modules' authors — How can you track your modules across the many different existing Linux distributions, helping *their* maintainers know when there are new versions available, probably including important fixes? If you change your modules' API or reorganize it in several submodules, what is the best way to notify them *before* bugs get filed? How does each distribution stand up regarding Perl and CPAN up-to-dateness and completeness?

Answering to these questions has proven a hard problem. Each distribution has different policies and ways of working with and presenting this information. The official CPAN ports page's section [Linux tracking] basically gave up on tracking Linux distributions in 2004, and it is frankly misleading to newcomers.

Gabor Szabo gathered a list of CPAN modules, checking for their presence and versions in several distributions and collections [Szabo]. From a quick glance, it is reasonably up to date (although it is clerly not generated on a daily or even weekly basis). This list is, of course, very useful and welcome for the purposes of this section — however, even when the distributions offer centralized tracking of the modules they package and offer to the users, operations apparently as trivial as following the packaged modules' *version numbers* is an interesting problem, as explained by Ricardo Signes[Signes]. Each distribution has different rules for interpreting the versioning[6] (with, of course, their own and very valid rea-

[6]Even going beyond what's programatically parseable, distributions don't always have a coherent way of translating the module *names* themselves — sometimes out of historic reasons, sometimes for technical reasons, sometimes for the distribution's policy (or lack of). CPAN *bundles* might be unbundled, or small but related modules might be bundled up together... But that topic, I believe, goes very quickly out of this paper's scope.

soning on why semantics are presented that way), and you might have to jump through hoops to get them correctly ordered.

3.2.2 A word about propietary systems

Of course, not everybody can use a free operating system such as Linux or the BSDs. For reasons many of us just cannot understand, some people even *want* to work with closed operating systems such as Microsoft Windows, MacOS X, or the historic Unixes.

Of course, propietary operating systems don't (and quite probably won't) offer this first level of support (certainly not outside their core components). For people using those systems, the best way out is to stick to the wonderful tools that have been created by the different communities. Nothing will cut the Perl cheese in Windows as the CPAN shell does, Solaris[7] will be a lonely place for PHP if you don't use PEAR, and developing Ruby on MacOS[8] without Gems will surely hurt.

My deepest sympathy to you guys.

4 Debian's pkg-perl group

4.1 How Debian maintainership works

Besides being a Linux distribution, Debian is a social project. Some like to define Debian as an ongoing experiment on a completely voluntary and (sometimes extremely) equalitarian organization. Debian is an organization of around 1000 official developers[9], all of them with the same rights regarding the project (which are, basically, getting a fancy @debian.org email address, being able to directly upload software to our *unstable* and *experimental* branches and being able to vote on the project's general resolutions). Each developer is responsible for the packages he/she *maintains*. For a long time, of course, several core infrastructure parts of the project have been developed and taken care of by groups of developers — But before 2004, group-maintainership was mostly limited to the areas that most obviously *required* it.

However, being a volunteer-run project, Debian has faced both the joys and the problems that stem straight out of its altruistic identity. The main problem we face is that, naturally, developers end up losing interest for the tasks they originally took on. Over the years, the Quality Assurance Team [Debian-QA] grew and wrote several tools to track the undermaintained packages and the maintainers that seem to be losing their steam. Their goal was, of course, to keep Debian's quality high — people are free to leave whenever they want to, but the project should not suffer because of it. Searches for MIA[10] developers following different strategies have been carried out at least since 2003 [Michlmayr, Troup], and they keep being refined today [Jaspert]. But maintainers should be interested in acting proactively — Many people in Debian, me included, see group maintainership, even for low-profile packages, as one of the best ways to prevent a demotivated maintainer from ending up becoming a liability to the whole project. If packages don't have only a single responsable person, they are much less prone to become undermaintained.

4.2 The pkg-perl group

In late 2003, a group of Debian Developers maintaining several Perl modules started coordinating in the `debian-perl@lists.debian.org` mailing list. By the beginning of 2004, Joachim Breitner sent out an announcement [pkg-perl started] inviting developers to join in a coordinated effort. Quoting his announcement,

[7] Yes, I know the nice guys at Sun Microsystems take pride at pointing out that OpenSolaris is truly free. It is, for most practical purposes. Solaris is, however, structurally still a historical, propietary-style Unix throughout. There is work underway to make installing and administering an OpenSolaris less daunting for people used to the Free Software way, and I truly hope we can soon install a free, robust and manageable OpenSolaris system. Still, as of today, it has a *long* way to learn about usability from the major Linux distributions.

[8] A similar note to 7 geared at Apple's strange Darwin/MacOS X combination.

[9] Plus several hundreds of unofficial developers, some of them working their way to become official, some of them just taking care of the bits they are interested in.

[10] Missing In Action — What, aren't we all fighting the same war?

At November 17th, a discussion about a common problem for Debian Perl developers was started on debian-perl@lists.debian.org. Most developers often realize that modules available on CPAN are not included in the Debian archive. This hinders the packaging of Perl applications and other modules.

After discarding the idea of automatically dumping all CPAN modules into the Debian archive, a collective effort to improve the packaging of Perl modules in Debian was proposed. This consists of creating new packages of needed Modules as well as of bugfixing and updating existing packages.

This seems to be necessary, as even many of the Perl modules included in the unstable distribution of Debian are outdated.

These thoughts lead to the founding of the Debian Perl Group, defined through the following goals:

- Adopt orphaned Perl module packages.
- Handle the RFP of Perl modules.
- Document and improve the usage of tools like `dh-make-perl`.
- Help with bugs in Perl packages.
- Keeping Perl packages in the Debian archive as up-to-date as possible.

I want to stress his second paragraph — Quite early in the process, we decided that, although we wanted to offer as much of CPAN as we could properly packaged for the Debian users, we did not want to blindly package all of CPAN. The whole point of having a *distribution* with formal QA processes is to have humans overseeing the process, ensuring the modules *are* buildable and usable at all times — And, very important, that all packages that form the stable distribution behave well with each other. During the development cycle, it is common for a module to change its API or to subtly break a dependency chain; some of the CPAN modules are basically proofs of concept or so domain specific they are used by just a small number of users, so although it *was* actually suggested, just blindly repackaging all of CPAN to fit the Debian structure, in short, never seriously worked on.

For our day-to-day operation, the pkg-perl group organizes and shares its work through a public SVN repository [pkg-perl SVN] where we keep the modules we package as well as the tools we use.

4.2.1 Current pkg-perl numbers and tools

As of today, the pkg-perl group is responsible for 327 packages [pkg-perl packages], out of 1303 that are part of the unstable branch — around one fourth of Debian's perl packages have been adopted by the group. Although there are formally almost 50 members in the group [pkg-perl], the number of active members is probably closer to 10. And, even though we are not free of bugs, the count is quite low given the amount of packages maintained — As of July 19, 2007, we have a total of 61 open bugs, only two of which are of release-critical priority. Out of the total 61 bugs, 8 are reported against the Mime-tools package, and most of those are tagged as upstream's decision.

4.2.2 DEHS: Keeping track of upstream versions

Of course, as part of the group's work is to track upstream development and keep Debian as up-to-date as possible regarding the CPAN modules, we heavily rely on the DEHS [DEHS]. All of our packages include the *magical* debian/watch file to keep track of new upstream versions, showing them in the group's packages page [pkg-perl packages]. The DEHS relies for its reports, however, on the ability to check for upstream versions periodically, and sometimes too often ends up reporting failures on some packages' up-to-dateness because of network timeouts, so it can not be taken as the only source to base our work on.

4.2.3 Evaluating and following the repository as a whole

As part of its natural growth, the pkg-perl group has not only packaged and adopted hundreds of Perl modules, but has also developed tools to keep the QA up to date. The scripts found at the `scripts/qa/` directory of our SVN tree [pkg-perl QA sources] are mostly the work of Gregor Hermann. They have only recently started running at our server (they were originally run by their author, who sent the reports by mail), and so far they are quite simple — But this is an area where improvement and new ideas should start flowing in soon.

The pkg-perl QA pages [pkg-perl QA] currently cover four areas:

build logs Periodically, all of our packages are rebuilt, to ensure they do not get broken due to changes in their dependencies. The full build logs, sorted by build result (success/failure) are available in the `qa/buildlogs` directory.

maintainers In Debian, a package can be *orphaned*, *adopted*, or sometimes even *hijacked*. When our group adopts a package, we must change the `Maintainer` and `Uploaders` fields prior to an upload. Our internal group convention (different groups do this in somewhat different fashions) mandates that the Maintainer should be set to the pkg-perl group, and the Uploaders should be the list of people specifically interested in this package's well being[11]. This script lists the modules for which our usual pattern does not hold.

versions Sometimes we fix a bug in our tree, or we prepare an upload for a new version (even more often when we take into account the non-Debian Developer members of the group). This script keeps track of the packages that still have to be uploaded — And while at it, tries again to check for upstream versions, given the problems described for the DEHS 4.2.2.

wnpp As stated in the pkg-perl goals 4.2, the pkg-perl group wants to take care of the problems regarding requested and orphaned Perl modules. As we will mention in footnote 12, however, the *Requests For Packaging* are sometimes unoperative. Sometimes, Perl packages can be orphaned for months or years before taken on. This script summarizes the information from WNPP so we can better tackle it, and reduce the number of undermaintained or missing Perl modules in Debian.

[11]If a package is adopted by a pkg-perl member who is not yet a Debian Developer, an official Debian Developer will have to do all the uploads. In this case, the official developer is not required to list him/herself in the Uploaders field — This is, in Debian language, called *sponsoring* a package.

4.3 dh-make-perl

Even if the Debian pkg-perl group, since its very formation, decided not to package every module on CPAN, we did want to provide our users with all the needed tools to quickly and easily integrate non-supported CPAN modules into the distribution — We understand Debian users will be more intimate with the Debian packaging and general system administration tools. Even though they can ask for any given Perl module to be packaged and included[12], we wanted our users to have the ability to integrate in their systems any module they require.

Back in October 2000, Paolo Molaro made the first upload of `dh-make-perl`. Since then, the package has been maintained by Ivan Kohler, Marc Brockschmidt and me. And although it was a completely separate project, given its goals are quite compatible to the group's, since October 2006, the package has been transferred to be group-maintained by the pkg-perl group. From the current package [dh-make-perl] description:

> Dh-make-perl will create the files required to build a Debian source package out of a perl package. This works for most simple packages and is also useful for getting started with packaging perl modules. Given a perl package name, it can also automatically download it from CPAN.

[12]The recommended way for requesting a module to be packaged for Debian is to file a RFP (*Request For Packaging*) bug on the `wnpp` (*Work Needed and Prospective Packaging*) pseudo-package in our BTS. One of the pkg-perl's goals, as stated in section 4.2, is to *handle the RFP for Perl modules*, but we have to admit that's more a statement of intent than anything else — Many RFPs just end up as unfulfilled requests, time out and get closed because no developer is willing to package the package

Dh-make-perl is an *ugly* script[13] that, given a Perl module (or even its distribution name in CPAN, which it downloads), prepares a valid Debian package, taking care to list all the dependencies that can be inferred from the module's metadata. Some work has even been done that would allow using `dh-make-perl` to be used to generate unofficial `APT` repositories. This is not as easy as it might sound — generating valid Debian packages requires digging up quite a bit of information. This consists of:

- The module's name and version
- A short and a long description
- Which infrastructure does it use for building (and how to query *that infrastructure* for the bits of information for filling this simplified list)
- Which other modules or packages does this one depend on in order to be used
- Which other modules or packages does this one depend on in order to be packaged
- Some dependency information regarding the current base Perl language and modules installed

This data is pulled from all over — Part of it comes from the `META.yml` file, if available, or from the `Makefile.PL` (in its different flavors, `MakeMaker`, `Module::Build`, `Module::Install`, from the module itself, even from the module's documentation. We have to predefine ways of building modules based on the different infrastructures... In short,`dh-make-perl` is an ugly, interesting beast.

5 Conclusion

This article was rather meant to draw a picture of our work than to reach any given conclusions. As you can see, we have gone a long way since we started building our infrastructure around four years ago — We just incorporated some new tools to the process, to make our work smoother and simpler. There is still a lot to be done, and one of the things we will most value is a closer interaction with our upstream authors.

On the other hand, there is much QA work that can flow from the distributions towards CPAN. We invite you to work with us with any requests you might have to better follow and enhance the development.

[13]It is so ugly that several of its past and present maintainers have publicly stated their desire to rewrite it from scratch. Lack of time (and lack of enough guts to face the beast) has prevented it from happening... But it *will* happen!

References

[Debian]	The Debian project `http://www.debian.org/`
[pkg-perl]	The Debian pkg-perl group `https://alioth.debian.org/projects/pkg-perl/`
[YAPC::EU]	YAPC::Europe 2007 (Vienna) `http://vienna.yapceurope.org/`
[Brocard]	Example implementation of Module::Packaged, by Leon Brocard `http://lists.debian.org/debian-perl/2003/11/msg00043.html`
[CPAN]	CPAN — Comprehensive Perl Archive Network `http://www.cpan.org/`
[CPAN Modules]	Full list of CPAN modules `http://www.cpan.org/modules/01modules.index.html`
[CPAN Authors]	Full list of CPAN authors `http://www.cpan.org/authors/00whois.html`
[CPAN Recent]	Updated list of CPAN's most recent uploads `http://www.cpan.org/modules/01modules.mtime.html`
[CPAN.pm]	CPAN.pm `http://search.cpan.org/~andk/CPAN-1.9102/lib/CPAN.pm`

[CPANPLUS] CPAN++ http://cpanplus.dwim.org/

[PEAR] PEAR: PHP Extension and Application Repository http://pear.php.net/

[Gems] Ruby Gems http://rubygems.org/

[PCS] The Python Cheese Shop http://www.python.org/pypi

[Cross] Dave Cross: Why Perl Advocacy Is A Bad Idea http://dave.org.uk/talks/advocacy.html

[Linux tracking] Tracking the Perl version provided by the different distributions http://www.cpan.org/ports/index.html#linux

[Szabo] CPAN Modules in Distributions http://www.szabgab.com/distributions/

[Signes] Ricardo Signes on distributions and version numbers http://use.perl.org/~rjbs/journal/33170?from=rss

[Debian-QA] The Quality Assurance team in Debian http://qa.debian.org/

[Michlmayr] Tracking inactive maintainers, Martin Michlmayr http://www.cyrius.com/talks/lsm-2003-mia/

[Troup] James Troup: Debian MIA check http://lists.debian.org/debian-devel-announce/2003/05/msg00006.html

[Jaspert] Jaspert: Handling of inactive Debian Accounts http://lists.debian.org/debian-devel-announce/2007/07/msg00004.html

[pkg-perl started] Debian Perl Group founding message http://lists.debian.org/debian-devel-announce/2004/debian-devel-announce-200401/msg00002.html

[pkg-perl SVN] pkg-perl SVN repository http://svn.debian.org/wsvn/pkg-perl

[pkg-perl packages] Packages overview for Debian Perl Group http://qa.debian.org/developer.php?login=pkg-perl-maintainers@lists.alioth.debian.org

[DEHS] Debian Watch Health Status http://dehs.alioth.debian.org/

[pkg-perl QA sources] http://svn.debian.org/wsvn/pkg-perl/scripts/qa/ pkg-perlgroup'sQAscriptssources

[pkg-perl QA] Debian Perl Group QA scripts http://pkg-perl.alioth.debian.org/qa/

[dh-make-perl] dh-make-perl package page http://packages.debian.org/unstable/devel/dh-make-perl