Grant Proposal:

FRDCSA Initial Release

Sign Petition On Table

Amount Requested:

$2250 +- $750

Synopsis

The FRDCSA consists of, among other things, over 1700 Perl5 modules in various stages of completion that operate in a tightly interwoven fashion. There are many entire systems for novel and important applications, especially from various subjects in Artificial Intelligence, as well as wrappers and APIs for other useful systems. Unfortunately, owing to their interdependence, lack of tests and documentation, need for a systematic renaming and deidentification - distributing them through CPAN has not yet been accomplished, although most of the software required to compute their distribution has been written. I seek a grant to facilitate their release either in majority or totality.
Benefits to the Perl Community

This grant IMHO would greatly improve the situation for Artificial Intelligence through Perl. But the benefits are not limited to pure theory, they are very practical - as these techniques, and the systems for which wrapping is provided, would then be instantly accessible to the entire Perl community. I do not expect the project to be understood from the getgo, however, as Perl programmers come across it, by searching CPAN for various tools, it should draw a small amount of attention - enough to bring the true benefits of the cohesive and interwoven system to the attention of Perl power users. As I am not a Perl expert, I am not familiar with all of the greatest aspects of Perl - however, I do know that there are many systems lacking for which I have labored over 10 years to provide the matching capabilities. Getting some attention for my project would be important to secure the goals of the system. The system, such as it is, is an attempt to implement a transfinite implementation of Hilbert's program. In this sense, it contains as its goal to solve all mathematical problems, which are thought to impose themselves on the real world. Goedelian incompleteness is not an obstacle, it is the cornerstone, based on associating increasingly complete systems of logic with ordinals ala Turing 1939.

How will this advance the release of Perl 6?

Interesting question that was not in the first template I filled out. I have often wondered whether this system is necessary to complete Perl6. I cannot comment precisely on exactly what it will do wrt the existing Perl 6 spec. However, if that spec is mutable, I think there are essential components. That is to say, if we can think of Perl 6 as being a more capable or "intelligent" version of Perl, then the availability of these tools will greatly enhance and furnish that goal. For instance, there is limited existing CPAN support for Semantic Web, Theorem Proving, or certain individual techniques from Natural Language Processing. One component of the FRDCSA is FRDCSAL which is a Perl based programming language where you write the code in English, and the code is translated to logic and executed. True, these are Perl5 modules, but when the automated refactoring system is complete, and the formal specification is complete - it will be possible to translate semi-automatically to Perl6. There is also a sub-project called Perform: which is a knowledge base about algorithms and data structures - which includes their complexity and all kinds of other features, intended as a devastatingly complete standard library for Perl6.

Deliverables

Many to all modules released.

Here is a very recent module list:

http://frdcsa.org/~andrewdo/projects/module-list.txt

For more information, please see the ancient description. Upon request I can regenerate the pages to reflect the current situation.

http://frdcsa.onshore.net/frdcsa

I have just this year started writing GUIs as frontends for various systems: Here is a page on that:

http://frdcsa.org/~andrewdo/projects/frdcsa-guis

There is also:

http://frdcsa.org
http://posithon.org
http://intranet.posithon.org

I am not familiar with milestone documents. The critical path is in the process of being computed, but won't be finished prior to the actual release. Here is an outdated document that sort of documents that:

http://files.meetup.com/970635/semweb.pdf

I have since added nested formulae and First Order w/Equality inferencing to FreeKBS2 - which makes the planning system much more capable - however, it has not been fully updated to use the new backend and so is not usable at the moment.

Also see number 13 at:

http://frdcsa.org/~andrewdo/projects/frdcsa-guis/

Project Details

The project is based primarily on Algorithmic Information Theory, and "Information Theoretic Limitations of Formal Systems". The basic idea is that, given a particular Turing machine, in order to accomplish tasks of increasing complexity, the software must ultimately grow in program length. Therefore, a necessary but insufficient condition for an "Artificial Intelligence" that has many

capabilities is that it is large. This means that one heuristic for finding existing such systems is that the system is large. This immediately makes one think of Perl, Debian and Emacs - which all check out as extremely capable systems. The system is based upon these, but will not be exclusive to them. However, as comprehensive as CPAN is, there are still elements missing.
Hence, we contemplate how to grow the system - there are two approaches - write software by ourselves - or find existing software. Both are taken. To find existing software, both existing large metasites (Sourceforge/Freshmeat/Google Code/etc) and active focused crawling for the location of new software. The Flossmole project releases data. To classify the data, the Debian tags and Sourceforge categories have been trained via an SVM text classifier to classify new entries according to these folksonomies. All this information goes into the CSO (Comprehensive Software Ontology). Focused crawling is achieved with the radar-web-search system, a multithreaded spider that searches several ply deep on websites for files that would seem to be useful. These are downloaded and go through an entire system which I cannot help but to compare to a digestive system. They are (semi-)automatically packaged for Debian using the Packager system.

For writing software, there are an equal number of sundry systems that expedite that process. For instance, all kinds of project templates and functions exist in the BOSS system. A particularly useful trick is ppi-convert-script-to-module.pl, which takes a script and converts it bit by bit to the standard Object Oriented Perl Module format. It is an example of a hand-coded automated refactoring that is going to be rearchitected into a deliberative automated refactoring system.
What is the goal of the project? The project aims to characterize all arguments for and against various beliefs, audit them for relative considerations, and present people with, if possible, a uniquely action guiding moral support intelligent agent. The basic goal of the project is to resolve problems that affect sentient beings in an equinaminous and peaceful manner. Many resource conflicts can be avoided through more rational behavior.

Inch-stones

There are several primary tasks for the project wrt CPAN. They are: Release, Refactor, Test and Document. The first three have corresponding automatic implementations: Release -> Task1, Refactor -> CodeMonkey, Test -> icodebase-testing. Documentation will be largely by hand and/or copied from my existing documents. Task1 is mostly complete. Refactor is 10-15% complete. Test is not really started. To fit within the time frame of the 3 month release schedule, I will focus only on Release (unless it happens faster than expected).
Release is divided into: Dependencies, Renaming, Deidentification, Testing. Dependencies are already computed but will be tidied up (have to reacquaint myself with the Task1 codebase), and possibly will now use FreeKBS2 logic programming to simplify the tasks. Testing will be only as necessary. Only about 2% of tests have been written. Renaming requires testing (to prevent large scale breakages), but is otherwise straightforward. Deidentification is tedious, however the Classify deidentification system is about 50-60% complete.

This should be enough to get the project into use and should generate future interactions which will ensure its completion.

Project Schedule

Project will take <= 3 months.

Can begin immediately upon receipt of initial funding segment.

Completeness Criteria

The project will be complete if most of the important systems of the FRDCSA are in CPAN and are not in a catastrophic condition. Given that there are 1700 modules, it is not reasonable (unless the automated refactoring system is finished, populated with adequate rules, and extremely accurate) to expect me to bring the code up to the quality standards that I myself do not know. Therefore, having them available in CPAN should be enough to attract attention to them, at which point in time what I need to do will be told to me rather than queried blindly and annoyingly of Perl experts. Many of them already function perfectly adequately or even superbly.