A Brief History of BioPerl [*]

Colin Crossman
Fellow, Duke University School of Law

Arti Rai
Professor, Duke University School of Law

December 2, 2005

Abstract

Large-scale open-source projects face a litany of pitfalls and difficulties. Problems of contribution quality, credit for contributions, project coordination, funding, and mission-creep are ever-present. Of these, long-term funding and project coordination can interact to form a particularly difficult problem for open-source projects in an academic environment.

BioPerl was chosen as an example of a successful academic open-source project. Several of the roadblocks and hurdles encountered and overcome in the development of BioPerl are examined through the telling of the history of the project. Along the way, key points of open-source law are explained, such as license choice and copyright.

The BioPerl project current status is then analyzed, and four different strategies typically employed by traditional open-source projects at this stage are analyzed as future directions. Strategies such as soliciting donations, securing grants, providing dual-licenses to enhance commercial interest, and the paid provision of support have all been employed in various traditional open-source projects with success, but each has drawbacks when applied to the academy. Finally, the construction of a successful long-term strategy for BioPerl, and other academic open-source projects, is proposed so that such projects can navigate the difficulties.

1 BioPerl's History

Perl is a unique programming language, well suited to the manipulation of textual strings [1]. The author, Larry Wall, released the first version of Perl on December 18th, 1987, while he was employed at Unisys [2]. Wall was given what seemed at the time to be an impossible task: connect and control twelve servers of varying types, each at a different site around the country. He developed Perl to address shortcomings in the prevalent tools of the time, such as *rn*, *patch*, *awk*, *warp*, and *sed*.

Released under an open source license, Perl quickly underwent many revisions, and in 1995, had progressed to version 5. Open source is a form of software distribution whereby the software users have access to the source code of the software, allowing them to review or modify it. In order to be called open source, the software must be released under a specific type of license, which gives the users permission not only to copy the source code, but also to redistribute the original software and any modifications. In some circumstances, under so-called viral licenses, the redistribution must also be made open source [3].

Perl was actually released under two separate open-source licenses: the GNU General Public License (GPL) and the Perl Artistic License. The GPL is a viral license, requiring that anyone who creates a derivative work of the GPL'd work to also make the derivative work available under the GPL [4]. The Perl Artistic License (Artistic License) was created by Wall, and is a much more permissive license than the GPL [5]. The Artistic License allows one to create commercially saleable derivative works with Perl, so long as the original work is made available free.

Wall incorporated several schema which serve to make Perl an extremely capable programming language, exceptional in that it is also very simple to learn [6]. The basic paradigm embodied by Perl is "There's more than one way to do it." Wall specifically and intentionally designed the language to include several ways to reach equivalent results [6].

The relative linguistic simplicity of Perl and the unique ease with which it can work with text-based information made it the natural choice for bioinformaticists. The researchers conducting large scale sequencing choose to store sequence information in literal notation rather than in a binary representation. The result was that genomic data was stored in many, often large, text files [7].

In 1996 a class of computer programmers from the Virtual School of Natural Sciences BioComputing Division (VSNS-BCD) was taught by Georg Fuellen [8]. The VSNS, loosely based at the University of Heidelberg in Germany, was an interesting venture of the Globe-wide Network Academy (GNA), an international distance learning program [9]. GNA began operations in 1993, and has maintained its presence as an aggregator and chartering organization for distance-learning programs, such as the VSNS. The VSNS was organized through the GNA, but was overseen by a board of directors comprising Dr. Marcus Speh, of the German High Energy Physics lab DESY, Dr. Peter Murray-Rust from Glaxo, Gustavo Glusman from the Weizmann Institute, Colman Reilly from Ireland's Trinity College, and Paul Hansen from the Together Foundation. Other programs the VSNS pursued include courses in protein structure, complex computing, and quantum mechanics.

The VSNS-BCD students started work on several biological problems. However, rather than take the approach of working on each new problem *de novo*, they built reusable modules for various aspects of their problems [10]. Since gene and protein sequence data is typically stored as human-readable text, Perl was the natural language to use.

Over the duration of the course, it became apparent to several of the students that the reusable code may be of use to general biocomputing problems outside of the class. Students Richard Reszick, Steve Brenner, Steve Chervitz, Jong Park, and Chris Dagdigian worked with Fuellen to coordinate these small programs, and the BioPerl project was born [11]. The first verifiable internet reference to BioPerl was on September 6th, 1996, on the VSNC-BCD email list set up to serve as the class's discussion board [12].

From that start, the BioPerl project was propagated by only a small number of developers, essentially working on their project without much outside help. Elsewhere, the Sanger center, the Whitehead Institute, Washington University and many others cooperating in the Human Genome Project were working on similar problems and generating similar solutions [7].

The problems facing the large institutions which made up the Human Genome Project stemmed from the fact that each of the various groups designed and deployed unique, redundant, proprietary software. In order to process the sequences produced by the wet labs, specialized software was created to handle the collection, annotation, quality control, and archiving of the collected sequence. Each of the participants created their own versions of these programs, and one institution's versions were not interoperable with other institutions [7, 13].

One software package did exist that attempted to address these intercommunication issues, Wisconsin GCG. Wisconsin GCG was developed in 1982 by the Genetic Computer Group and stored its information in text-based 'gcg' files still popular today for certain types of sequence data storage. However, Wisconsin GCG was primarily a sequence analysis platform, and not a sequence conversion platform.

The general lack of conversion tools forced many members of the Human Genome Project laboratories to write their own, which they initially did independently. Lincoln Stein was working with the Human Genome Project at Cold Spring Harbor National Laboratory. Stein developed a package known as BoulderIO, which acted to ameliorate several of these transport difficulties present in the Human Genome Project [7]. Ewan Birney at the Sanger Institute had developed sequence and align modules for use internally at EMBL. Ian Korf at the Washington University Geneome Sequencing Center, working on gene discovery and

genome features in *C. elegans*, had generated a set of Perl utilities for genome analysis, which included comparatively advanced gene markup capabilities.

In 1998, Lincoln Stein, Ewan Birney, and Ian Korf contacted the BioPerl developers, and each offered their work and their expertise to the project [14, 15, 16]. Each of the submissions from this triad of luminaries was available to the BioPerl project "free, of course", as epitomized by Ian Korf [17].

The influx and subsumption of these co-developed projects rapidly moved BioPerl forward. The active set of modules was quickly expanded to include an enhanced sequence manipulation feature set as well as interoperability with several databases like NCBI-BLAST and GenBank at the National Library of Medicine and the EMBL databases [18].

The integration of a BLAST module which could accurately and faithfully communicate with the NCBI-BLAST service allowed BioPerl to reach a much wider audience. Nearly everyone who works with DNA or proteins uses BLAST in some capacity. One specific drawback to NCBI-BLAST is that its output is inconsistently formatted, necessitating a very well made parser. Though creating a BLAST parser is similar in character to translating file formats, the inconsistencies in BLAST are less well known. Anyone working with BLAST would have to either spend considerable time working through BLAST's inconsistencies to write their own parser, or attempt to use a parser they obtained off the Internet. While there were several parsers available at the time, none were integrated with a feature set as rich as BioPerl's, and not all of the parsers were actively supported and developed.

Along with the quickened pace of development came a greater managerial burden. To ease this burden, several BioPearl team members have acted as "chief coordinator". The first coordinator was Fuellen, by virtue of his coordination of the various students to begin the project. Sometime in 1998, Steve Chervitz assumed the central role as chief. Steve stepped down in 2000, and passed the torch to Ewan Birney, who had just moved to the European Bioinformatics Institute [19, 20]. Birney's ascension began a more focused period of development of BioPerl. For one, Birney was involved heavily with the Ensembl project, and had adopted BioPerl as the architecture of Ensembl's automated gene annotation system. Second, about this time many new people became interested in the project, and Birney's leadership helped to organize the growing volume of submissions.

Chris Dagdigian took on the roll of system administrator, managing the details of the BioPerl internet presence. He personally acquired the Internet domain names for the BioPerl web sites, and began running a server to establish the BioPerl Internet presence. The initial BioPerl site was a 486-class linux machine constructed of discarded parts from the

Genetics Institute. Dagdigian ran the server out of his home, connecting to the Internet over an ISDN line he personally paid for [21]. Later, when the project outgrew that modest server, Chris secured a state-of-the-art Alpha server through a joint donation from Compaq's Pharmaceutical Sales group and High Performance Technical Computing group [21]. In addition, Dagdigian's employer, the Genetics Institute (currently the Wyeth Andover Research Facility), donated collocation support in the form of server room space, electricity, and Internet bandwidth to the new server, allowing for a much more stable and capable Internet presence. The Genetics Institute especially found this donation to be of mutual benefit they were able to advertise their support to the relevant communities, and reap the rewards of increased respect and higher quality employee candidates [21].

Several incremental test-releases were generated between 1998 and 2002, including version 0.7 on March 5th, 2001, which comprised a major rewrite and reworking. The 0.7 release also helped to solidify the main BioPerl team. Hilmar Lapp of the Genomics Institute of the Novartis Research Foundation and Jason Stajich from the Duke University Center for Human Genetics, signed on and began an overhaul of BioPerl's core routines [22, 23]. In the cases of Stajich and Lapp, both entered the BioPerl community and rose to the top through the traditional Open-Source mechanism: their contributions were good. Over several months, both Stajich and Lapp stood out from the crowd by volunteering to undertake difficult and important projects, and, more importantly, delivering good solutions. They also worked to bring the BioPerl package to a greater variety of computing platforms, such as Microsoft Windows, broadening the potential user base. Others, such as Heikki Levashalo from EMBL worked on adding new functionality to the package.

During this time, the completion of the Human Genome Project and the increasing scope of other bioinformatics projects helped to foment widespread adoption of the BioPerl packages. The team generated documentation and tutorials so that newcomers could easily integrate the toolkit into their projects.

Though BioPerl was in general use during this time, it wasn't until the first stable release on March 18, 2002, of BioPerl Version 1.0, that the software was truly ready for end-users [24]. Due in great part to the main team, and in particular to Ewan Birney, Jason Stajich, Hilmar Lapp, Heikki Levashalo, Steve Chervitz and Chris Dagdigian, BioPerl Version 1.0 offered the biological sciences a toolkit unmatched in quality and completeness. Coupled with the release of Version 1.0, the team also published a paper detailing the project and the toolkit, ensuring widespread adoption [46].

Since the completion of Release 1.0 of BioPerl, several related projects have arisen to extend the idea of BioPerl to other areas of computing. Projects incorporating other pro-

gramming languages, such as BioJava, BioPython and BioRuby, enhance the ability of researchers unskilled in Perl to incorporate these methods. Projects such as BioCORBA and BioXML have arisen to ease the creation of programs to pass biological data from one system to another. Together, these projects are grouped under the moniker Bio*, and are all available from the Open Bioinformatics Foundation [26]. Though all of the Bio* projects deal with similar information, they differ in focus since the contributors to each project had different backgrounds. For example, while BioPerl is centered on sequence data, BioPython is more specialized to deal with small molecules and their interactions with proteins [14].

—- Place Figure 1 here —-

The BioPerl project might never have been, however, were it not for the underlying principle which the contributors all implicitly accepted. They all freely gave their work to the project, and in turn, the entirety of the BioPerl project is available under the Perl Artistic License, an open source license which guarantees open access to the underlying source code, and does not inhibit commercial utilization of the software [5].

The BioPerl team has built on that initial success by fastidiously maintaining their project. Though BioPerl was initially adopted by many groups due to its specific strengths, it has been able to grow further due to the ongoing support that the team has put into it. In every way the ideals of Open Source have been integrated into other aspects of the project the BioPerl mailing lists are nexus between the users of the project and the project managers. Users with questions find those lists to be amenable places to receive answers, often from other users, but also from the BioPerl team itself. Sometimes, exchanges that begin with a confused user begin the process of adding a new feature into the BioPerl set, such as the introduction of an RNA fold parser into the BioPerl system [27].

Another aspect of BioPerl's enduring success is an aggressive education campaign that members of the core team engage in. The team gave a BioPerl "bootcamp" to an audience of academic and industry researchers in Montreal during the summer of 2004 [28]. Jason Stajich has given several BioPerl tutorials at the National Institutes of Environmental Health Sciences, as well as in academic institutions. Lincoln Stein uses BioPerl in his annual Genome Informatics course given at the Cold Spring Harbor National Laboratory.

The widespread acceptance of BioPerl can be seen on a practical level by examining job postings more and more employers are listing BioPerl experience as a specific attribute, and more and more job-seekers are highlighting their BioPerl knowledge [**?**]. This trend will only continue as the BioPerl toolkit is adopted by academic and institutional players, exhibiting the traditional network effects of technological adoption.

Additionally, some bioinformatics software companies are beginning to incorporate BioPerl into their products. In particular, a SciTegic produces a product called Pipeline Pilot that incorporates the BioPerl system [30]. Pipeline Pilot seeks to simplify the flow of data, easing the user's need to understand how to work with BioPerl, thus bringing the powerful functionality of the BioPerl tools to a vastly increased user base [31]. SciTegic also occationally contributes back to the BioPerl project, for instance if they were to discover a bug in the BioPerl system [31].

BioPerl may be unique among open-source bioinformatics projects, however. Those working in the field note that, while open-source is a good way for code in this region to be released, they don't often find much of that code of use [32]. Part of this stems from the fact that much of this code is written to solve very specific problems that other researchers are unlikely to encounter.

Opensource projects, and indeed all software projects, possess several layers of utility and portability: system-level, user-level, and problem-level portability. System-level portability refers to the ability of a project to operate on a variety of platforms - such as Microsoft Windows or Linux. Without system-level portability, a project written on one platform will not operate on another platform, limiting its potential market. User-level portability refers to the ability of the target audience to adopt the project. Problem-level portability refers to the ability of the project to address a wide spectrum of problems.

BioPerl exhibits all three levels of portability. Written in Perl, it is readily usable on nearly any platform, and therefore has broad system-level portability. Also, being Perl, it potentially has a broad user-level portability, as Perl has a quick learning curve. Finally, BioPerl is essentially a general purpose package, and includes a wide array of utilities for biological and bioinformatic analysis. Utilities including databases tuned for bioinformatic information, analysis routines for genomics, proteomics, and evolutionary studies, and the ability to add new tools as needed. All the tools included in BioPerl, and its general-purpose nature, cause it to have a tremendously broad problem-level portability.

Posessing all three aspects of portability, BioPerl, and other projects like it, can generate a fair amount of network effects, over and above those created by more specific projects. This also has the added bonus that the contributor pool is similarly expanded, increasing the number and quality of code contributions.

2 BioPerl's Future

One problem with BioPerl, and indeed any Open Source project, is that its success comes with the seeds of its own demise. As a project becomes larger, the problems of coordination become progressively more acute. Formerly, team members Steve Chervitz and Ewan Bierny stepped forward to undertake the management of the project [19, 20]. However, BioPerl has reached the stage where continued support has become a full-time proposition. Myriad developers are submitting new material for inclusion in later versions of the package, and those must be tested for fitness. Similarly, any bug-fixes submitted must be sorted to remove the wheat from the chaff.

The coordination problems are actually made more difficult by the very people who have been supporting the project to this point. In a real sense, the entirety of the BioPerl project exists in the heads of the core BioPerl team. Later developers just don't have the experience wrestling with the project to have a complete understanding of the system. As time goes on, the members of the core team progress in their respective careers, slowly limiting their ability to invest time and effort in the project.

Further, at this stage, the project is still the brain-child of the core team. In order to off-load the day-to-day needs of the project, the core would also need to transfer operational control. Though they could be parsimonious in their selection, there would be no guarantee that, over time, the new director will share all or even most of the ideals that the core team embodied. This is a very difficult step, but one that must be undertaken by any core members who wish to lessen their involvement. Founders of some projects have found ways to maintain their involvement. For example, Linus Torvalds maintains direct operational control of Linux, and Larry Wall with Perl itself. However, both of these people have been given expansive rein by their employers (Linus Torvalds is currently at Transmeta, and Larry Wall is at O'Reilly Publishing) to maintain those projects, and have turned that maintenance into careers.

Once the "who" aspect of the coordination problem is solved, a second aspect, perhaps more difficult than the first, arises. Anyone hired in such a capacity would probably have to forego other career opportunities, such as proceeding along the tenure track, and devote full time and energy to BioPerl. That would require remuneration. The as-yet unsolved problem with Open Source projects of this scope is related to payment. By its very nature, Open Source is free. How does a free project raise enough funds to support itself, once it grows beyond the hobby sphere and into the enterprise sphere? Several methods of funding exist, with the most prominent including soliciting donations, securing government grants,

dual licenses for commercial users, and the provision of knowledgeable support. None of these methods offer a perfect solution, however, and many have significant drawbacks.

While being the most tenuous, the soliciting of donations actually has limited success in supporting many projects. The projects usually best served by this method of support are generally projects with a small number of contributors [33]. However, a handful of the largest Open Source projects do successfully take advantage of donation based funding, such as the Apache Software Foundation and SourceForge.net [34, 35]. Funding received this way depends on the altruism of the public, and as sponsor-based radio and television have shown, it can be very difficult to raise these funds. Additionally, though both Apache and SourceForge.net have made stunning cases for the success of donation-based funding, and both share some surface characteristics with BioPerl, neither example is well analogized to the BioPerl situation.

The Apache Software Foundation, like BioPerl, started as a group of developers collaborating to put together a cohesive project [36]. Apache differs from BioPerl in one simple, but important, way, that functions to confound BioPerl's adoption of a similar donation model. Apache is a fundamental Internet infrastructure technology. The potential growth of Apache is unhindered by the boundaries of a small concerned community. Therefore, the Apache Software Foundation has a manifestly larger population to depend upon for donation support. In contrast, BioPerl is targeted at the small, nascent community of Bioinformaticists and Bioinformaticians.

SourceForge.net is even further afield from BioPerl than Apache. The SourceForge.net site maintains itself off of a combination subscription/donation model. However, their "products" are not only open source software but hosting services and compile farm capabilities, which are highly amenable to this form of model. Another method that the BioPerl team could consider is that of government grants. For instance, from 2002-2005, the NIH had available a special grant vehicle specifically for the maintenance of Bioinformatics software [37, 38, 39, 40, 41]. This grant expired on February 25, 2005, and while a new grant replaced it that day, the expiration highlights one of the most problematic attributes of this funding method [42]. Federal grants have an evanescent nature, and their continued existence is dependant on the whims and budgetary constraints of the granting organizations. Even if the grant itself continues, there is no guarantee any individual grantee will be successful in reapplying for the grant. Still, given the demonstrated utility of BioPerl, the supplementation of their funding stream with this method could prove to be highly beneficial.

A third funding method commonly used in successful Open Source projects is the offering of paid support. RedHat has utilized this particular strategy to great effect [43].

This support can take many forms, and indeed, the BioPerl team has a history of this. For instance, heavily involved team members can be called upon to give lectures and training seminars to programmers. The potential drawbacks of this method include the continued time constraints placed on the core team. If the entire purpose of obtaining funding is to delegate the day-to-day maintenance of the project to allow the team members to curtail their involvement, being called upon to give seminars whenever and wherever they are required seems unlikely to solve that problem. Paying the team members to give the seminars may be a solution, but that necessarily diminishes the amount of funds which can be channeled back into the organization. Also, while hiring several people to give seminars and training sessions full time may be an answer, it will have to be evaluated for cost-efficacy.

Finally, the project could implement a dual-licensing regime, similar to that notably adopted by SendMail [44]. SendMail, like Apache, is a network infrastructure technology, providing an email transmission protocol. In fact, SendMail was the original email transport program to offer SMTP (Simple Mail Transfer Protocol), the most commonly used email transport protocol today. SendMail, Inc. administers both a for-profit wing that develops the saleable version of SendMail, and the open-source wing that provides the open-source version. While the open-source version provides the basic functionality of the saleable version, purchasers of the software have access to a wide array of closed-source utilities not available to the open-source version. In this way, SendMail has wielded the power of the free market to support the continued development of open software.

BioPerl may be able to benefit heavily from a dual-license method; however, it is fraught with difficulties. One major manifestation of the problems that BioPerl may encounter would be the assignment of Intellectual Property rights to any custom-designed solutions. If BioPerl retains the rights to any innovations, will a Biotech company pay BioPerl to devise a solution to a problem, if they know their competitor may then simply purchase the solution from BioPerl? Conversely, if BioPerl is contracted to produce a solution for a company, and then assign the rights to that company, and stumbles across errors in a different area of the BioPerl code, who owns the fix to that disparate area? However, these issues are those which may be resolved by creative contracting, and may be solvable.

A typical problem frustrating the dual-license avenue is the problem of actually collecting enough rights in BioPerl to be able to offer a dual license. Since BioPerl was designed as a collaborative effort, with each participant retaining rights in their submissions, no one entity holds all the rights to the BioPerl code. Oftentimes, concentrating the rights into a single entity presents its own collective action problem contributors need to be contacted and convinced to sign over their rights. Holdouts may occur, either for money, or to prevent the

perceived privatization of their code. In BioPerl's case, however, the problem of holdouts is minimal, since all of BioPerl's code has been released under the Perl Artistic License, which is very permissive in allowing licensees to profit from the sale of the code.

The problem of a coalescence of rights is not unprecedented in the Open Source community. Linux itself may face a similar problem with the rise of the GPL v.3. Contributions to the Linux kernel, the prime component of the operating system, have been under the GPL v.2, modified to remove the language authorizing later versions of the GPL to supersede GPL v.2. If the Linux community decides that the GPL v.3 is sufficiently improved, it will be unable to "upgrade" to the GPL v.3 by fiat or democratic directive, by the simple fact that no one knows who contributed what code anymore. The only way to upgrade will be to recreate the entire kernel under the GPL v.3. Linux has the advantage of having a truly massive user and contributor base, so this process, though not trivial, will not be prohibitively difficult. BioPerl, at present, lacks such a huge community of contributors, and also lacks the restrictions of the GPL [45].

One bright spot in the above morass of unsatisfactory solutions is the negligible cost of infrastructure. Since the costs involved in maintaining a viable Internet presence are very inexpensive and declining rapidly, the monies necessary to support that presence will be a declining share of the overall expense. Indeed, certain members of the BioPerl core group have intimated that they are willing to put up their own time and funds on a continuing basis to ensure that there is an Internet infrastructure for BioPerl as long as there is a BioPerl. While this does not abrogate the need for the project to become totally self-sufficient, alleviation of that minor tension can only help the project's long term viability.

In light of the benefits and drawbacks of the above funding options, BioPerl should consider a mlange of several methods. The strongest concern that was raised by both the BioPerl team and the corporate users of the BioPerl system was the existance of a corporate structure to manage the project [31, 46]. Fortunatly, BioPerl already has created a corporate structure in place, the Open Bioinformatics Foundation. The first step would be to secure an NIH grant to bootstrap that organization. After that, the organization would continue to solicit donations through this period. The organization would build itself to offer support for the software to the community as a whole, including training seminars. By combining these funding methods, the drawbacks of each are lessened. Using the NIH grant, full time employees could be supported to give the seminars, which bring in the operating income (or, depending on accounting requirements, the reverse). Further, by creating a network of users, they enhance the possibility that the project will become inculcated fully into the field, and thereby enhance the chance that the field would support a production spin-off to

provide expensive, tailored solutions.

By all accounts, the rise of BioPerl has had a tremendous benefit on the field of Bioinformatics. The fact that they have reached the delicate crossroads of open-source funding is not indicative of any weakness in BioPerl, but rather that the BioPerl project has been wildly successful.

References

[*] All web links cited in this article were accessed and confirmed on August 30, 2005.

[1] Perl, an acronym like most things originating in the Unix world, stands for "Practical Extraction and Reporting Language".

[2] Larry Wall, Tom Christiansen & Jon Orwant, *Programming Perl*, Chapter 27, 645-649 (3d ed. O'Reilly 2000).

[3] Bruce Perens, "Open Sources: Voices from the Open Source Revolution," (1st ed. O'Reilly 1999).

[4] <**http://www.gnu.org/licenses/gpl.html**>.

[5] <**http://www.perl.com/pub/a/language/misc/Artistic.html**>.

[6] Lorrie Cranor, *An Interview with Larry Wall*, Crossroads 1.2 (Winter 1994) available at: <**http://www1.acm.org/crossroads/xrds1-2/lwall.html**>.

[7] Lincoln Stein, "How Perl saved the Human Genome Project," 2(1) The Perl Journal (Summer 1996).

[8] <**http://www.techfak.uni-bielefeld.de/bcd/original-welcome.html**>.

[9] <**http://www.gnacademy.org/**>.

[10] Charles Krueger, Software Reuse, 24(2) ACM Computing Surveys 131-183 (June 1992) available online at <**http://sunnyday.mit.edu/16.355/kruger.pdf**>.

[11] "A bit of history", from the BioPerl.org website. <**http://www.bioperl.org/Thanks.shtml**>.

[12] <**http://bioperl.org/pipermail/bioperl-l/1996-September/002618.html**>

[13] Interview with Lincoln Stein, 5/24/2005.

[14] <**http://bioperl.org/pipermail/bioperl-l/1998-March/002833.html**>.

[15] <**http://bioperl.org/pipermail/bioperl-l/1998-March/002834.html**>.

[16] <**http://bioperl.org/pipermail/bioperl-l/1998-May/002836.html**>.

[17] <**http://bioperl.org/pipermail/bioperl-l/1998-April/002835.html**>.

[18] SF Altschul, et al., Basic Local Alignment Search Tool, 215 J. Mol. Bio 403-410 (1990).

[19] <**http://bioperl.org/pipermail/bioperl-l/2000-January/003373.html**>.

[20] <**http://bioperl.org/pipermail/bioperl-l/2000-January/003374.html**>.

[21] Interview with Chris Dagdigian, 4/13/2005.

[22] <**http://bioperl.org/pipermail/bioperl-l/2000-March/003428.html**>.

[23] <**http://bioperl.org/pipermail/bioperl-l/2000-May/003487.html**>.

[24] <**http://bioperl.org/pipermail/bioperl-l/2002-March/007513.html**>.

[25] Stajich, J., et al., "The Bioperl Toolkit: Perl Modules for the Life Sciences," 12(10) Genome Research 1611-1618 (October 2002).

[26] <**http://www.open-bio.org/**>.

[27] <**http://bioperl.org/pipermail/bioperl-l/2003-December/014156.html**>.

[28] <**http://bioperl.org/pipermail/bioperl-l/2004-April/015481.html**>.

[29] Electric Genetics job posting of 10/15/2003, <**http://bioinformatics.org/forums/forum.php?forum_id=2171**>.

[30] SciTegic (http://www.scitegic.com/) is a subsidiary of Accelrys, Inc. (http://www.accelrys.com/).

[31] Telephone interview with Dr. Scott Markel, SciTegic, 11/17/2005.

[32] Interview with Dave and Jane Richardson on 2/10/2005, and Michael Eisen on 5/18/2005.

[33] Clive Thompson, The BitTorrent Effect, 13.01 Wired 150 (2005) available at <**http://www.wired.com/wired/archive/13.01/bittorrent.html**>.

[34] <**http://www.apache.org/foundation/faq.html**>.

[35] <**http://sourceforge.net/donate/**>.

[36] How Apache Came To Be, online resource, available at <**http://httpd.apache.org/ABOUT_APACHE.html**>.

[37] NIH Grant PA-02-141, Continued Development and Maintenance of Bioinformatics and Computational Biology Software, Release: 7/26/2002; Expires 2/25/2005.

[38] NIH Grant #5R01GM068630-02 to Joel R. Stiles for MCell/DReAMM: A Microphysiological Modeling Environment.

[39] NIH Grant #1R01GM070923-01 to John C. Doyle for Continued Support and Development of SBML.

[40] NIH Grant #5R01HG003040-02 to Gregg A. Helt for DAS2: A Distributed Genome Annotation Sequence.

[41] NIH Grant #5R01GM070557-02 to Wah Ghiu for Icosahedral Particle Reconstruction Software.

[42] NIH Grant PAR-05-057, Continued Developmentand Maintenance of Software, Released: 2/25/2005, Expires: 9/14/2007.

[43] RedHat, Inc., Why Subscriptions? <**http://www.redhat.com/about /mission/business_model.html**>.

[44] Heather Meeker, Dual-Licensing Open Source Business Models, 3(4) LinuxWorld 26-28 (April 6, 2005).

[45] Ingrid Marson, GPL 3 not expected to split free-software world, ZDNet News, March 25, 2005 available at <**http://news.zdnet.com/2100-3513_22-5637496.html**>.

[46] Interview with Jason Stajich, 5/17/2005.

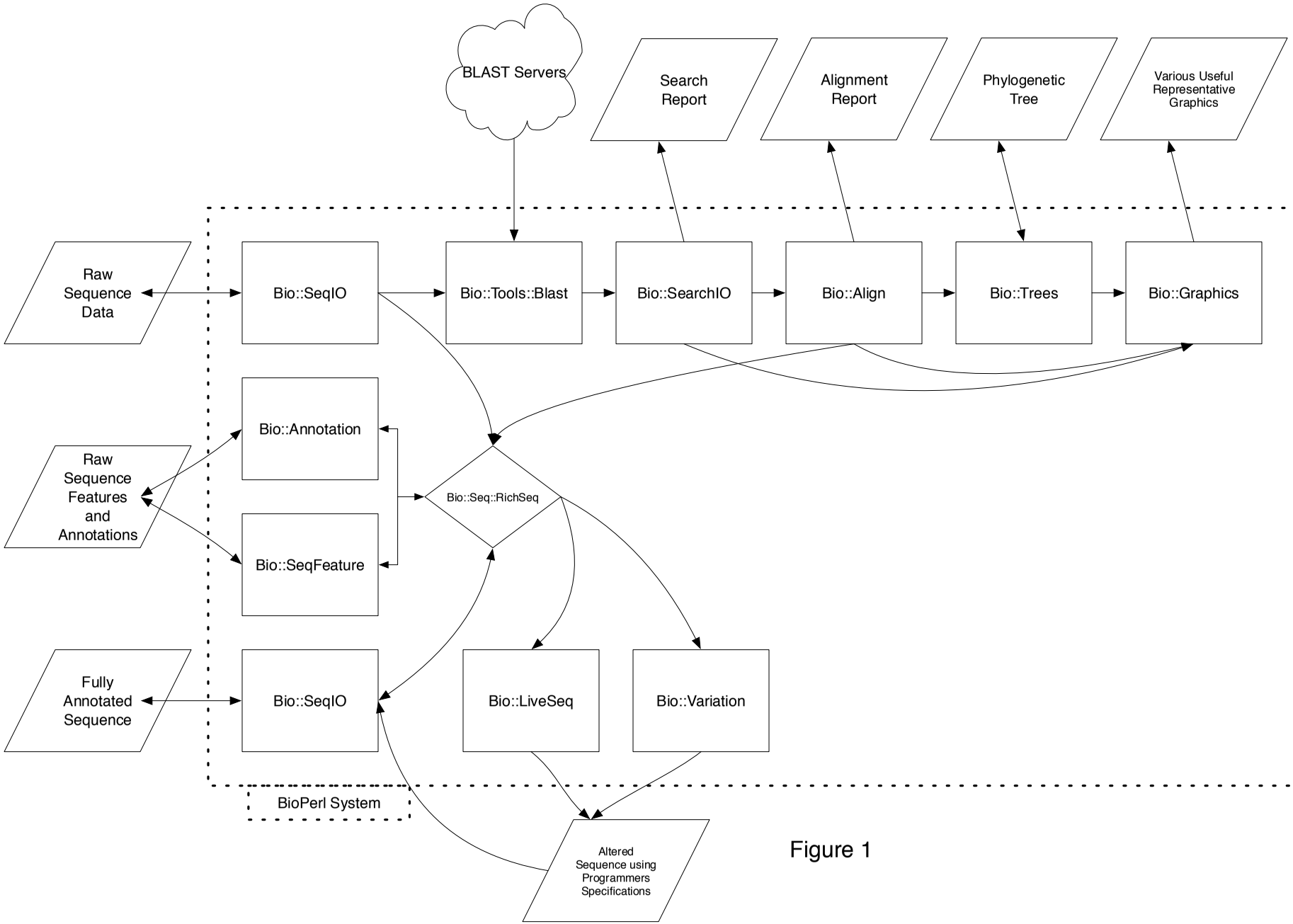
BLAST Servers
Search Report
Alignment Report
Phylogenetic Tree
Various Useful Representative Graphics
Raw Sequence Data
Bio::SeqIO
Bio::Tools::Blast
Bio::SearchIO
Bio::Align
Bio::Trees
Bio::Graphics
Bio::Annotation
Raw Sequence Features and Annotations
Bio::Seq::RichSeq
Bio::SeqFeature
Fully Annotated Sequence
Bio::SeqIO
Bio::LiveSeq
Bio::Variation
BioPerl System
Altered Sequence using Programmers Specifications

Figure 1