Software

An integrated computational pipeline and database to support whole-genome sequence annotation

CJ Mungall*, S Misra[†‡], BP Berman[†], J Carlson[§], E Frise[§], N Harris[‡§], B Marshall[†], S Shu[†‡], JS Kaminker[†‡], SE Prochnik[†‡], CD Smith[†‡], E Smith[†‡], JL Tupy[†‡], C Wiel[†‡], GM Rubin*[†‡§] and SE Lewis[†‡]

Addresses: *Howard Hughes Medical Institute, University of California, Berkeley, CA 94720, USA. [†]Department of Molecular and Cellular Biology, Life Sciences Addition, University of California, Berkeley, CA 94720-3200, USA. [‡]FlyBase-Berkeley, University of California, Berkeley, CA 94720-3200, USA. [§]Genome Sciences Department, Lawrence Berkeley National Laboratory, One Cyclotron Road, Berkeley, CA 94720, USA.

Correspondence: CJ Mungall. E-mail: cjm@fruitfly.org

Published: 23 December 2002

Genome ***Biology*** 2002, **3(12)**:research0081.1–0081.11

The electronic version of this article is the complete one and can be found online at http://genomebiology.com/2002/3/12/research/0081

Received: 25 October 2002
Accepted: 28 November 2002

(Print ISSN 1465-6906; Online ISSN 1465-6914)

Abstract

We describe here our experience in annotating the *Drosophila melanogaster* genome sequence, in the course of which we developed several new open-source software tools and a database schema to support large-scale genome annotation. We have developed these into an integrated and reusable software system for whole-genome annotation. The key contributions to overall annotation quality are the marshalling of high-quality sequences for alignments and the design of a system with an adaptable and expandable flexible architecture.

Rationale

The information held in genomic sequence is encoded and highly compressed; to extract biologically interesting data we must decrypt this primary data computationally. This generates results that provide a measure of biologically relevant characteristics, such as coding potential or sequence similarity, present in the sequence. Because of the amount of sequence to be examined and the volume of data generated, these results must be automatically processed and carefully filtered.

There are essentially three different strategies for whole-genome analysis. The first is a purely automatic synthesis from a combination of analyses to predict gene models. The second aggregates analyses contributed by the research community that the user is then required to integrate visually on a public website. The third is curation by experts using a full trail of evidence to support an integrated assessment. Several groups charged with rapidly providing a dispersed community with genome annotations have chosen the purely computational route; examples are Ensembl [1] and the National Center for Biotechnology Information (NCBI) [2]. Approaches using aggregation adapt well to the dynamics of collaborative groups which are focused on sharing results as they accrue; examples are the University of California Santa Cruz (UCSC) genome browser [3] and the Distributed Annotation System (DAS) [4]. For organisms with well established and cohesive communities the demand is for carefully reviewed and qualified annotations; this approach was adopted by three of the oldest genome-community databases, SGD for *Saccharomyces cerevisiae* [5], ACeDB for *Caenorhabditis elegans* (documentation, code and data available from anonymous FTP servers at [6]) and FlyBase for *Drosophila melanogaster* [7].

We decided to examine every gene and feature of the *Drosophila* genome and manually improve the quality of the

annotations [8]. The prerequisites for this are: first, a computational pipeline and a database capable of both monitoring the pipeline's progress and storing the raw analysis; second, an additional database to provide the curators with a complete, compact and salient collection of evidence and to store the annotations generated by the curators; and third, an editing tool for the curators to create and edit annotations based on this evidence. This paper discusses our solution for the first two requirements. The editing tool used, Apollo, is described in an accompanying paper [9].

Our primary design requirement was flexibility. This was to ensure that the pipeline could easily be tuned to the needs of the curators. We use two distinct databases with different schemata to decouple the management of the sequence workflow from the sequence annotation data itself. Our long-term goal is to provide a set of open-source software tools to support large-scale genome annotation.

Sequence datasets

The sequence datasets are the primary input into the pipeline. These fall into three categories: the *D. melanogaster* genomic sequence; expressed sequences from *D. melanogaster*; and informative sequences from other species.

Release 3 of the *D. melanogaster* genomic sequence was generated using bacterial artificial chromosome (BAC) clones that formed a complete tiling path across the genome, as well as whole-genome shotgun sequencing reads [10]. This genomic sequence was 'frozen' when, during sequence finishing, there was sufficient improvement in the quality to justify a new 'release'. This provided a stable underlying sequence for annotation.

In general, the accuracy and scalability of gene-prediction and similarity-search programs is such that computing on 20 million base (Mb) chromosome arms is ill-advised, and we therefore cut the finished genomic sequence into smaller segments. Ideally, we would have broken the genome down into sequence segments containing individual genes or a small number of genes. Before the first round of annotation, however, this was not possible for the simple reason that the position of the genes was as yet unknown. Therefore, we began the process of annotation using a non-biological breakdown of the sequence. We considered two possibilities for the initial sequence segments, either individual BACs or the segments that comprise the public database accessions. We rejected the use of individual BAC sequences and chose to use the GenBank accessions as the main sequence unit for our genomic pipeline because the BACs are physical clones with physical breaks, while the GenBank accession can subsequently be refined to respective biological entities. At around 270 kilobases (kb), these are manageable by most analysis programs and provide a convenient unit of work for the curators. To minimize the problem of genes straddling these arbitrary units we first fed the BAC sequences into a lightweight version of the full annotation pipeline that estimated the positions of genes. We then projected the coordinates of these predicted genes from the BAC clones onto the full-arm sequence assembly. This step was followed by the use of another in-house software tool to divide up the arm sequence, trying to simultaneously optimize two constraints: to avoid the creation of gene models that straddle the boundaries between two accessions; and to maintain a close correspondence to the pre-existing Release 2 accessions in GenBank/EMBL/DDBJ [11-13]. During the annotation process, if a curator discovered that a unit broke a gene, they requested an appropriate extension of the accession prior to further annotation. In hindsight we have realized that we should have focused solely on minimizing gene breaks because further adjustments by GenBank were still needed to ensure that, as much as possible, genes remained on the same sequence accession.

To re-annotate a genome in sufficient detail, an extensive set of additional sequences is necessary to generate sequence alignments and search for homologous sequences. In the case of this project, these sequence datasets included assembled full-insert cDNA sequences, expressed sequence tags (ESTs), and cDNA sequence reads from *D. melanogaster* as well as peptide, cDNA, and EST sequences from other species. The sequence datasets we used are listed in Figure 1 and described more fully in [8].

Software for task monitoring and scheduling the computational pipeline

There are three major infrastructure components of the pipeline: the database, the Perl module (named Pipeline), and sufficient computational power, allocated by a job-management system. The database is crucial because it maintains a persisting record reflecting the current state of all the tasks that are in progress. Maintaining the jobs, job parameters and job output in a database avoids some of the inherent limitations of a file-system approach. It is easier to update, provides a built-in querying language and offers many other data-management tools that make the system more robust. We used a MySQL [14] database to manage the large number of analyses run against the genome, transcriptome and proteome (see below).

MySQL is an open-source 'structured query language' (SQL) database that, despite having a limited set of features, has the advantage of being fast, free, and simple to maintain. SQL is a database query language that was adopted as an industry standard in 1986. An SQL database manages data as a collection of tables. Each table has a fixed set of columns (also called fields) and usually corresponds to a particular concept in the domain being modeled. Tables can be cross-referenced by using primary and foreign key fields. The database tables can be queried using the SQL language, which

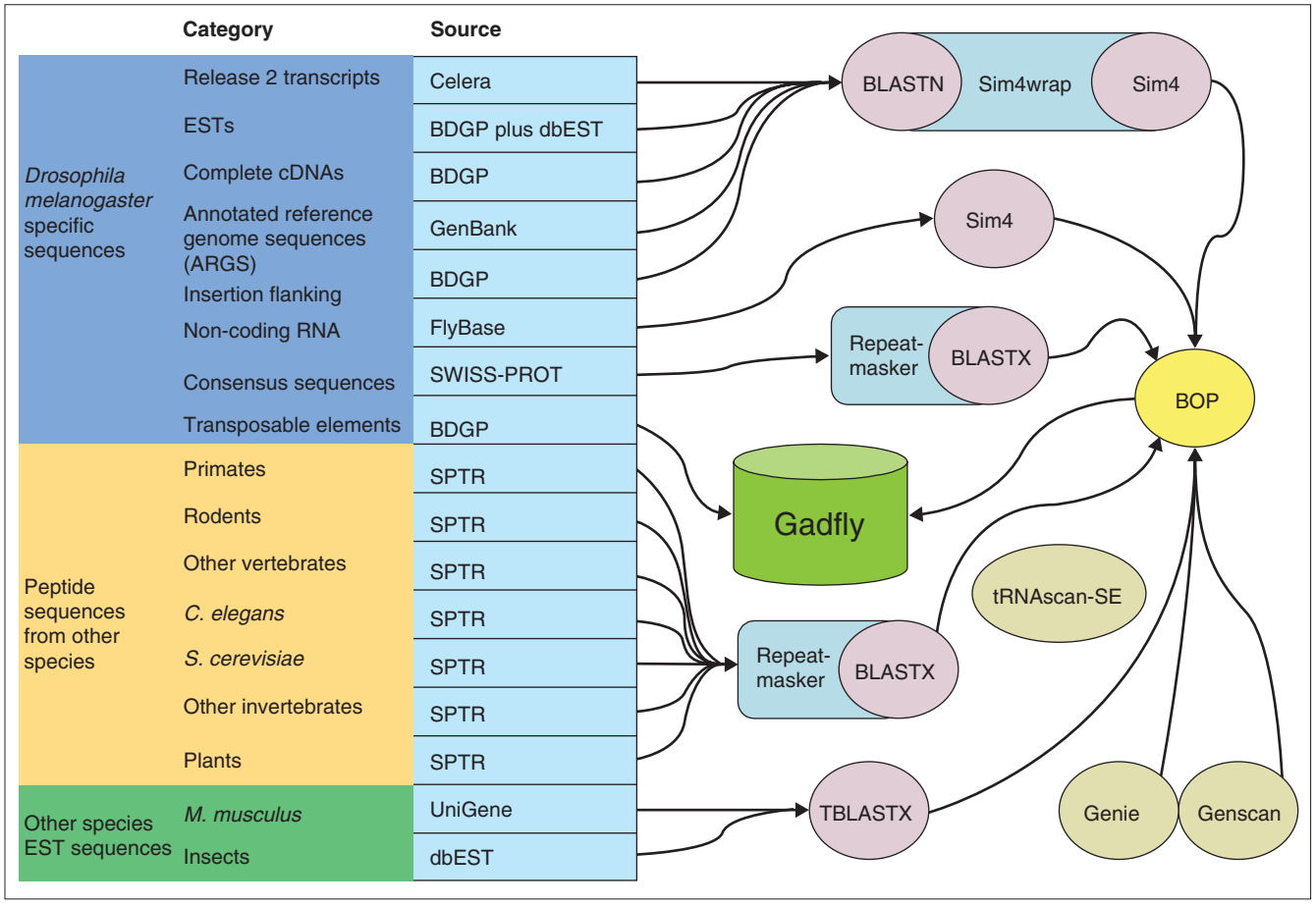

Figure 1
Gadfly data sources and analyses. This figure provides an overview of the pipeline analyses that flow into the central annotation database (Gadfly) and are provided to the curators for annotation. The *D. melanogaster*-specific datasets (dark blue) are one of the following: nucleic acids, peptides (from SPTR: SWISS-PROT/TrEMBL/TrEMBLNEW [31]), or transposable elements (the source of the sequences are listed in the light-blue column). The nucleic acids are aligned using sim4 and the peptides using BLASTX. The transposable elements are the product of a more detailed analysis [46] and their coordinates were recorded directly in Gadfly. The peptide datasets from other species (yellow) were obtained from SWISS-PROT and aligned using BLASTX. We used TBLASTX to translate (in all six frames) and align the rodent UniGene [47] and insect ESTs from dbEST [48] (green). For *ab initio* predictions on the genomic sequence we used Genie [42], Genscan [43] and tRNAscan-SE [44]. BOP was used to filter BLAST and sim4 results and parse all the results to output GAME XML; the results were recorded in Gadfly by loading the XML into the database.

allows the dynamic combination of data from different tables [15]. A collection of these tables is called a database schema, and a particular instantiation of that schema with the tables populated is a database. The Perl modules provide an application programmer interface (API) that is used to launch and monitor jobs, retrieve results and support other interactions with the database.

There are four basic abstractions that all components of the pipeline system operate upon: a sequence, a job, an analysis and a batch. A 'sequence' is defined as a string of amino or nucleic acids held either in the database or as an entry in a FASTA file (usually both). A 'job' is an instance of a particular program being run to analyze a particular sequence, for example running BLASTX to compare one sequence to a peptide set is considered a single job. Jobs can be chained together. If job A is dependent on the output of job B, then the pipeline software will not launch job A until job B is complete. This situation occurs, for example, with programs that require masked sequence as input. An 'analysis' is a collection of jobs using the same program and parameters against a set of sequences. Lastly, a 'batch' is a collection of analyses a user launches simultaneously. Jobs, analyses and batches all have a 'status' attribute that is used to track their progress through the pipeline (Figure 2).

The three applications that use the Perl API are the pipe_launcher script, the flyshell interactive command line interpreter, and the internet front end [16]. Both pipe_launcher and flyshell provide pipeline users with a variety of powerful ways to launch and monitor jobs, analyses and batches. These tools are useful to those with a basic

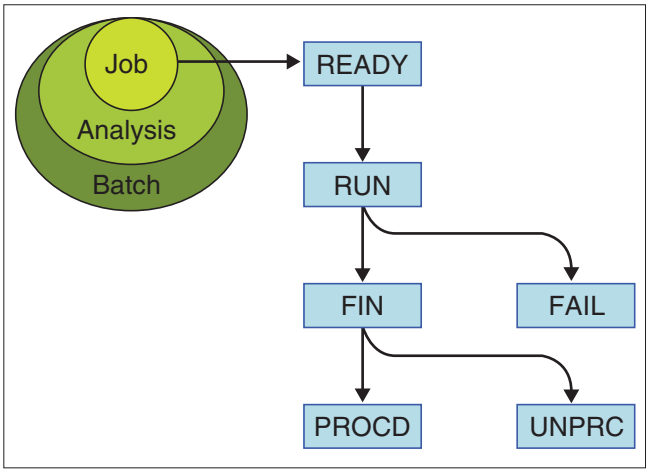

Figure 2
Pipeline job management. The pipeline database tracks the status of jobs, analyses and batches. As indicated by the green ovals, a batch is a collection of analyses, and an analysis is a set of jobs. A job is a single execution of a program on a single sequence (for example, BLASTX similarity searching of a unit of genomic sequence). All three have a current task status. The slowest running in the set dictates the status of an analysis and a batch. Thus, in terms of analyses, the analysis status is the same as the status of the slowest job in that analysis, and for batches, the status is the same as the slowest analysis in that batch. The allowed values for the status attribute are READY, RUN, FIN, PROCD, UNPRC and FAIL. With respect to jobs, READY means the jobs are ready to be sent to the pipeline queue, RUN means the jobs are on the queue or being run, FIN means the jobs have run but have not yet been processed by BOP to extract the results from the raw data, UNPRC generally means there was an error in the processing step, FAIL means there was an error in job execution, and PROCD means the jobs have run and been processed by BOP.

understanding of Unix and bioinformatics tools, as well as those with a good knowledge of object-oriented Perl. The web front end is used for monitoring the progress of the jobs in the pipeline.

The pipe_launcher application is a command-line tool used to launch jobs. Users create configuration files that specify input data sources and any number of analyses to be performed on each of these data sources, along with the arguments for each of the analyses. Most of these specifications can be modified with command line options. This allows each user to create a library of configuration files for sending off large batches of jobs that can be altered with command-line arguments when necessary. Pipe_launcher returns the batch identifier generated by the database to the user. To monitor jobs in progress, the batch identifier can be used in a variety of commands, such as 'monitor', 'batch', 'delete-batch' and 'query_batch'.

The flyshell application is an interactive command-line Perl interpreter that presents the database and pipeline APIs to the end user, providing a more flexible interface to users who are familiar with object-oriented Perl.

The web front end allows convenient, browser-based access for end users to follow the status of analyses. An HTML form allows users to query the pipeline database by job, analysis, batch or sequence identifier. The user can drill down through batches and analyses to get to individual jobs and get the status, raw job output and error files for each job. This window on the pipeline has proved a useful tool for quickly viewing results.

Once a program has successfully completed an analysis of a sequence, the pipeline system sets its job status in the database to FIN (Figure 2). The raw results are recorded in the database and may be retrieved through the web or Perl interfaces. The raw results are then parsed, filtered and stored in the database and the job's status is set to PROCD. At this point a GAME (Genome Annotation Markup Elements) XML (eXtensible Markup Language [17]) representation of the processed data can be retrieved through either the Perl or web interfaces.

Analysis software

In addition to carrying out computational analyses, a critical function of the pipeline is to screen and filter the output results. There are two primary reasons for this: to increase the efficiency of the pipeline by reducing the amount of data that computationally intensive tasks must process, and to increase the signal-to-noise ratio by eliminating results that lack informative content. We now describe the auxiliary programs we developed for the pipeline.

Sim4wrap

Sim4 [18] is a highly useful and largely accurate way of aligning full-length cDNA and EST sequences against the genome [19]. Sim4 is designed to align nearly identical sequences and if dissimilar sequences are used, the results will contain many errors and the execution time will be long. To circumvent this problem, we split the alignment of *Drosophila* cDNA and EST sequences into two serial tasks and wrote a utility program, sim4wrap, to manage these tasks. Sim4wrap executes a first pass using BLASTN, using the genome sequence as the query sequence and the cDNA sequences as the subject database. We run BLASTN [20,21] with the '-B 0' option, as we are only interested in the summary part of the BLAST report, not in the high-scoring pairs (HSPs) portion where the alignments are shown. From this BLAST report summary sim4wrap parses out the sequence identifiers and filters the original database to produce a temporary FASTA data file that contains only these sequences. Finally we run sim4 again using the genomic sequence as the query and the minimal set of sequences that we have culled as the subject.

Autopromote

The *Drosophila* genome was not a blank slate because there were previous annotations from the Release 2 genomic sequence [22]. Therefore, before the curation of a chromosome

arm began, we first 'autopromoted' the Release 2 annotations and certain results from the computational analyses to the status of annotations. This simplified the annotation process by providing an advanced starting point for the curators to work from.

Autopromotion is not a straightforward process. First, there have been significant changes to the genome sequence between releases. Second, all of the annotations present in Release 2 must be accounted for, even if ultimately they are deleted. Third, the autopromotion software must synthesize different analysis results, some of which may be conflicting. Autopromote resolves conflicts using graph theory and voting networks.

Table 1 lists the programs and parameters that were used for the analysis of the genomic sequence and peptide analysis.

Berkeley Output Parser filtering

We used relatively stringent BLAST parameters to preserve disk space and lessen input/output usage and left ourselves the option of investigating more deeply later. In addition, we used the Berkeley Output Parser (BOP) with the following adjustable parameters to process the BLAST alignments and remove HSPs that did not meet our annotation criteria.

Minimum expectation is the required cutoff for a HSP. Any HSP with an expectation greater than this value is deleted; we used 1.0 x e^{-4} as a cutoff.

Remove low complexity is used to eliminate matches that primarily consist of repeats; such sequences are specified as a repeat word size - that is, the number of consecutive bases or amino acids - and a threshold. The alignment is compressed using Huffman encoding to a bit length, and hits where all HSP spans have a score lower than this value are discarded.

Maximum depth permits the user to limit the number of matches that are allowed in a given genomic region. This parameter applies to both BLAST and sim4. The aim is to avoid excess reporting of matches in regions that are highly represented in the aligned dataset, such as might arise between a highly expressed gene and a non-normalized EST library. The default is 10 overlapping alignments. However, for sim4, we used a value of 300 to avoid missing rarely expressed transcripts.

Eliminate shadow matches is a standard filter for BLAST that eliminates 'shadow' matches (which appear to arise as a result of the sum statistics). These are weak alignments to the same sequence in the same location on the reverse strand.

Table 1

Software used in the analysis pipeline

Program	Data source	Parameters
RepeatMasker [41]	Transposable elements	-parallel 2 -nolow -keepmasked
Sim4wrap	Gadfly Release2	BLAST:
	ESTs	-B0 -V10000 -E1e-10
	cDNA sequence reads	Sim4
	BDGP cDNAs	-A 4
	GenBank cDNAs	
	ARGS	
Sim4 [18]	Non-coding RNAs	-A 4
WU-BLASTX [20]	Fly	-B800 -V800 -Z300000 -E1e-10
	Community reports	
WU-BLASTX	Non-fly	-B800 -V800 -Z300000 -E1e-10
		SEG+XNU
WU-TBLASTX	dbEST (insect)	-B200 -V200 -Z300000 -E1e-10
	UniGene (rodent)	SEG+XNU
Genie [42]		
Genscan [43]		
tRNAscan-SE [44]		
McPromoter [45]		
ClustalW [34]		version 1.8
Align		
InterProScan [32]		

Sequential alignments reorganizes BLAST matches if this is necessary to ensure that the HSPs are in sequential order along the length of the sequence. For example, a duplicated gene may appear in a BLAST report as a single alignment that includes HSPs between a single portion of the gene sequence and two different regions on the genome. In these cases the alignment is split into two separate alignments to the genomic sequence.

Sim4 filtering

Our primary objective in using sim4 was to align *Drosophila* ESTs and cDNA sequences only to the genes that encoded them, and not to gene-family members, and for this reason we applied stringent measures before accepting an alignment. For sim4 the filtering parameters are as follows.

Score is the minimum percent identity that is required to retain an HSP or alignment; the default value is 95%.

Coverage is a percentage of the total length of the sequence that is aligned to the genome sequence. Any alignments that are less than this percentage length are eliminated; we required 80% of the length of a cDNA to be aligned.

Discontinuity sets a maximum gap length in the aligned EST or cDNA sequence. The primary aim of this parameter is to identify and eliminate unrelated sequences that were physically linked by a cDNA library construction artifact.

Remove poly(A) tail is a Boolean to indicate that short terminal HSPs consisting primarily of runs of a single base (either T or A because we could not be certain of the strand) are to be removed.

Join 5′ and 3′ is a Boolean operation and is used for EST data. If it is true, BOP will do two things. First, BOP will reverse complement any hits where the name of the sequence contains the phrase '3prime'. Second, it will merge all alignments where the prefixes of the name are the same. Originally this was used solely for the 5′ and 3′ ESTs that were available. However, when we introduced the internal sequencing reads from the *Drosophila* Gene Collection (DGC) cDNA sequencing project [23] into the pipeline this portion of code became an alternative means of effectively assembling the cDNA sequence. Using the intersection of each individual sequence alignment with the genome sequence a single virtual cDNA sequence was constructed.

Another tactic for condensing primary results, without removing any information, is to reconstruct all logically possible alternative transcripts from the raw EST alignments by building a graph from a complete set of overlapping ESTs. Each node comprises the set of spans that share common splice junctions. The root of the graph is the node with the most 5′ donor site. It is, of course, also possible to have more than one starting point for the graph, if there are overlapping nodes with alternative donor sites. The set of possible transcripts is the number of paths through this tree(s). This analysis produced an additional set of alignments that augmented the original EST alignments.

External pipelines

Of the numerous gene-prediction programs available, we incorporated only two into our pipeline. This was because some of these programs are difficult to integrate into a pipeline, some are highly computationally expensive and others are only available under restricted licenses.

Rather than devoting resources to running an exhaustive suite of analyses, we asked a number of external groups to run their pipelines on our genomic sequences. We received results for three of the five chromosome arms (2L, 2R, 3R) from Celera Genomics, Ensembl and NCBI pipelines. These predictions were presented to curators as extra analysis tiers in Apollo and were helpful in suggesting where coding regions were located. However, in practice, human curators required detailed alignment data to establish biologically accurate gene structures and this information was only available from our internal pipeline.

Hardware

As an inexpensive solution to satisfy the computational requirements of the genomic analyses we built a Beowulf cluster [24] and utilized the portable batch system (PBS) software developed by NASA [25] for job control. A Beowulf cluster is a collection of processor nodes that are interconnected in a network and the sole purpose of these nodes and the network is to provide processor compute cycles. The nodes themselves are inexpensive off-the-shelf processor chips, connected using standard networking technology, and running open-source software; when combined, these components generate a low-cost, high-performance compute system. Our nodes are all identical and use Linux as their base operating system, as is usual for Beowulf clusters.

The Beowulf cluster was built by Linux Networx [26] which also provided additional hardware (ICE box) and Clusterworx software to install the system software and control and monitor the hardware of the nodes. The cluster configuration used in this work consisted of 32 standard IA32 architecture nodes, each with dual Pentium III CPUs running at 700 MHz/1 GHz and 512 MB memory. In addition, a single Pentium III-based master node was used to control the cluster nodes and distribute the compute jobs. Nodes were interconnected with standard 100BT Ethernet on an isolated subnet with the master node as the only interface to the outside network. The private cluster 100BT network was connected to the NAS-based storage volumes housing the data and user home directories with Gigabit ethernet. Each node had a 2 GB

swap partition used to cache the sequence databases from the network storage volumes. To provide a consistent environment, the nodes had the same mounting points of the directories as all other BDGP Unix computers. The network-wide NIS maps were translated to the internal cluster NIS maps with an automated script. Local hard disks on the nodes were used as temporary storage for the pipeline jobs.

Job distribution to the cluster nodes was done with the queuing system OpenPBS, version 2.3.12 [25]. PBS was configured with several queues and each queue having access to a dynamically resizable overlapping fraction of nodes. Queues were configured to use one node at a time, either running one job using both CPUs (such as the multithreaded BLAST or Interpro motif analysis) or two jobs using one CPU each for optimal utilization of the resources. Because of the architecture of the pipeline, individual jobs were often small but tens of thousands of them may be submitted at any given time. Because the default PBS first-in/first-out (FIFO) scheduler, while providing a lot of flexibility, does not scale up beyond about 5,000-10,000 jobs per queue, the scheduler was extended. With this extension the scheduler caches jobs in memory if a maximum queue limit is exceeded. Job resource allocation was managed on a per queue basis. Individual jobs could only request cluster resources based on the queue they were submitted to and each queue was run on a strict FIFO basis. With those modifications PBS was scaled to over 100,000 jobs while still permitting higher-priority jobs to be submitted to a separate high-priority queue.

Storing and querying the annotation results: the Gadfly database

A pipeline database is useful for managing the execution and post-processing of computational analyses. The end result of the pipeline process is streams of prediction and alignment data localized to genomic, transcript or peptide sequences. We store these data in a relational database, called Genome Annotation Database of the Fly (Gadfly). Gadfly is the second of the two database schemata used by the annotation system and will be discussed elsewhere.

We initially considered using Ensembl as our sequence database. At the time we started building our system, Ensembl was also in an early stage of development. We decided to develop our own database and software, while trying to retain interoperability between the two. This proved difficult, and the two systems diverged. While this was wasteful in terms of redundant software development, it did allow us to hone our system to the particular needs of our project. Gadfly remains similar in architecture and implementation details to Ensembl. Both projects make use of the bioPerl bioinformatics programming components [27-29].

The core data type in Gadfly is called a 'sequence feature'. This can be any piece of data of biological interest that can be localized to a sequence. These roughly correspond to the types of data found in the 'feature table' summary of a GenBank report. Every sequence feature has a 'feature type' - examples of feature types are 'exon', 'transcript', 'protein-coding gene', 'tRNA gene', and so on.

In Gadfly, sequence features are linked together in hierarchies. For instance, a gene model is linked to the different transcripts that are expressed by that gene, and these transcripts are linked to exons. Gadfly does not store some sequence features, such as introns or untranslated regions (UTR), as this data can be inferred from other features. Instead Gadfly contains software rules for producing these features on demand.

Sequence features can have other pieces of data linked to them. Examples of the kind of data we attach are: functional data such as Gene Ontology (GO) [30] term assignments; tracking data such as symbols, synonyms and accession numbers; data relevant to the annotation process, such as curator comments [8]; data relevant to the pipeline process, such as scores and expectation values in the case of computed features. Note that there is a wealth of information that we do not store, particularly genetic and phenotypic data, as this would be redundant with the FlyBase relational database.

A core design principle in Gadfly is flexibility, using a design principle known as generic modeling. We do not constrain the kinds of sequence features that can be stored in Gadfly, or constrain the properties of these features, because our knowledge of biology is constantly changing, and because biology itself is often unconstrained by rules that can be coded into databases. As much as possible, we avoid built-in assumptions that, if proven wrong, would force us to revisit and explicitly modify the software that embodies them.

The generic modeling principle has been criticized for being too loosely constrained and leading to databases that are difficult to maintain and query. This is a perceived weakness of the ACeDB database. We believe we have found a way round this by building the desired constraints into the program components that work with the database; we are also investigating the use of ontologies or controlled vocabularies to enforce these constraints. A detailed discussion of this effort is outside the scope of this paper and will be reported elsewhere.

Figure 3 shows the data flow in and out of Gadfly. Computational analysis features come in through analysis pipelines - either the Pipeline, via BOP, or through an external pipeline, usually delivered as files conforming to some standardized bioinformatics format (for example, GAME XML, GFF).

Data within Gadfly is sometimes transformed by other Gadfly software components. For instance, just before curation of a chromosome arm commences, different computational

analyses are synthesized into 'best guesses' of gene models, as part of the autopromote software we described earlier.

During the creation of Release 3 annotations, curators requested data from Gadfly by specifying a genomic region. Although this region can be of any size, we generally allocated work by GenBank accessions. Occasionally, curators worked one gene at a time by requesting genomic regions immediately surrounding the gene of interest. Gadfly delivers a GAME XML file containing all of the computed results and the current annotations within the requested genomic region. The curator used the Apollo editing tool to annotate the region, after which the data in the modified XML file was stored in Gadfly.

The generation of a high-quality set of predicted peptides is one of our primary goals. To achieve this goal, we needed a means of evaluating the peptides and presenting this assessment to the curators for inspection, so that they might improve the quality of the predicted peptides iteratively. Every peptide was sent through a peptide pipeline to assess the predicted peptide both quantitatively and qualitatively. Where possible, we wanted to apply a quantifiable metric, requiring a standard against which we could rate the peptides. For this purpose we used peptides found in SPTREAL (E. Whitfield, personal communication), a carefully reviewed database of published peptide sequences, for comparison to our predicted proteins. SPTREAL is composed of 3,687 *D. melanogaster* sequences from the SWISS-PROT and TrEMBL protein databases [31] and provides a curated protein-sequence database with a high level of annotation, a minimal level of redundancy and the absence of any hypothetical or computational gene models. Our program, PEP-QC, performed this crucial aspect of the annotation process and is described below. In cases where a known peptide was

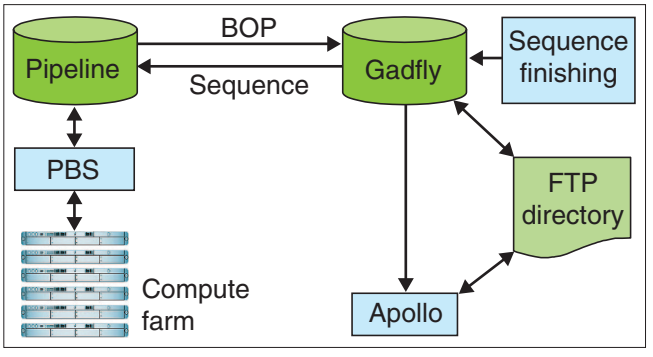

Figure 3
Pipeline dataflow. Finished genomic sequence is deposited in Gadfly, and then fed to the pipeline database, which manages jobs, dispatching them to the compute farm via PBS. When a job finishes, the pipeline database stores the output. BOP filters this output and exports GAME XML to Gadfly. A cycle of annotation consists of curators loading GAME XML into Apollo, either directly from Gadfly or from a data directory. Modified annotations are then written to a directory and loaded into Gadfly.

unavailable, we used a qualitative measure to evaluate the peptide. The peptide pipeline provided a BLASTP analysis with comparisons to peptides from other model organism genome sequences and InterProScan [32] analysis for protein-family motifs to enable the curators to judge whether the biological properties of the peptide were reasonable.

Each annotation cycle on a sequence may affect the primary structure of the proteins encoded by that sequence and these changes must therefore trigger a reanalysis of the edited peptides. Whereas the genomic pipeline is launched at distinct stages, on an arm-by-arm basis, the peptide pipeline is run whenever a curator changes a gene model and saves it to the Gadfly database. To rapidly identify whether the peptide sequence generated by the altered gene model has also changed, the database uniquely identifies every peptide sequence by its name and its MD5 checksum [33]. The MD5 checksum provides a fast and convenient way of determining whether two sequences are identical. To determine whether a peptide sequence has been altered is a simple comparison of the prior checksum to the new checksum, allowing us to avoid using compute cycles in reanalyzing sequences that have not changed.

PEP-QC generates both summary status codes and detailed alignment information for each gene and each peptide. ClustalW [34] and showalign [35] are used to generate a multiple alignment from the annotated peptides for the gene and the corresponding SPTREAL peptide or peptides. In addition, brief 'discrepancy' reports are generated that describe each SPTREAL mismatch clearly. For instance, an annotated peptide might contain any or all of the mismatches in Table 2 (in this example, CG2903-PB is the initial FlyBase annotation and Q960X8 is the SPTREAL entry).

The quality assessments produced by the peptide pipeline need to be available to the curators for inspection during annotation sessions so that any corrections that are needed can be made. Curators also need to access other relevant FlyBase information associated with a gene in order to refine an annotation efficiently. We developed automatically generated 'mini-gene-reports' to consolidate this gene data into a single web page. Mini-gene-reports include all the names and synonyms associated with a gene, its cytological location and accessions for the genomic sequence, ESTs, PIR records and Drosophila Gene Collection [23] assignments, if any. All of these items are hyperlinked to the appropriate databases for easy access to more extensive information. All literature references for the gene appear in the reports, with hyperlinks to the complete text or abstracts. The mini-gene-reports also consolidate any comments about the gene, including amendments to the gene annotation submitted by FlyBase curators or members of the *Drosophila* community. The mini-gene-reports can be accessed directly from Apollo, or searched via a web form by gene name, symbol, synonym (including the FlyBase unique identifier, or FBgn) or

Table 2

Example of PEP-QC output

Peptide position	SPTR*REAL* position	Discrepancy description
M1	M1..E88	Amino-terminal insertion: Q960X8 contains an additional stretch of 88 amino acids
CG2903-PB	**Q960X8**	
M163	K163	Internal substitution of one amino acid
CG2903-PB	**Q960X8**	
G1533..D1537	P1580..Q158	Carboxy-terminal replacement of five amino acids with 10 amino acids
CG2903-PB	**Q960X8**	

For each discrepancy we show the sequence affected and the position on the sequence for both Gadfly and SWISS-PROT peptides. The peptide sequence identifiers are in bold.

genomic location. A web report, grouped by genomic segment and annotator, is updated nightly and contains lists of genes indexed by status code and linked to their individual mini-gene-reports.

Other integrity checks

Before submission to GenBank a number of additional checks are run to detect potential oversights in the annotation. These checks include confirming the validity of any annotations with open reading frames (ORF) that are either unusually short (less than 50 amino acids) or less than 25% of the transcript length. In the special case of known small genes, such as the *Drosophila* immune response genes (DIRGs) [36], the genome annotations are scanned to ensure that no well-documented genes have been missed. Similarly, the genome is scanned for particular annotations to verify their presence, including those that have been submitted as corrections from the community, or are cited in the literature, such as tRNA, snRNA, snoRNA, microRNA or rRNA genes documented in FlyBase. If the translation start site is absent, an explanation must be provided in the comments. Annotations may also be eliminated if annotations with different identifiers are found at the same genome coordinates or if a protein-coding gene overlaps a transposable element, or a tRNA overlaps a protein-coding gene. Conversely, duplicated gene identifiers that are found at different genome coordinates are either renamed or removed. A simple syntax check is also carried out on all the annotation symbols and identifiers. Known mutations in the sequenced strain are documented and the wild-type peptide is submitted in place of the mutated version.

The BDGP also submits to GenBank the cDNA sequence from the DGC project. Each of these cDNA clones represents an expressed transcript and it is important to the community that the records for these cDNA sequences correctly correspond to the records for the annotated transcripts in both GenBank and FlyBase. This correspondence is accomplished via the cDNA sequence alignments to the genome described previously. After annotation of the entire genome was completed these results were used to find the intersection of cDNA alignments and exons. A cDNA was assigned to a gene when the cDNA overlapped most of the gene exons and the predicted peptides of each were verified using a method similar to PEP-QC.

Public World Wide Web interface

We provide a website for the community to query Gadfly. This allows queries by gene, by genomic or map region, by Gene Ontology (GO) assignments or by InterPro domains. As well as delivering human-readable web pages, we also allow downloading of data in a variety of computer-readable formats supported by common bioinformatics tools. We use the GBrowse [37] application, which is part of the GMOD [38] collection of software for visualization and exploration of genomic regions.

Software engineering

The main software engineering lesson we learned in the course of this project was the importance of flexibility. Nowhere was this more important than in the database schema. In any genome, normal biology conspires to break carefully designed data models. Among the examples we encountered while annotating the *D. melanogaster* genome were: the occurrence of distinct transcripts with overlapping UTRs but non-overlapping coding regions, leading us to modify our original definition of 'alternative transcript'; the existence of dicistronic genes, two or more distinct and non-overlapping coding regions contained on a single processed mRNA, requiring support for one to many relationships between transcript and peptides; and *trans*-splicing, exhibited by the *mod*(*mdg4*) gene [39], requiring a new data model. We also needed to adapt the pipeline to different types and qualities of input sequence. For example, to analyze the draft sequence of the repeat-rich heterochromatin [40], we needed to adjust the parameters and datasets used, but also to develop an entirely new repeat-masking approach to facilitate gene finding in highly repetitive regions. We are now in the process of modifying the pipeline to exploit comparative genome sequences more efficiently. Our intention is to continue extending the system to accommodate new biological research situations.

Improvements to tools and techniques are often as fundamental to scientific progress as new discoveries, and thus the sharing of research tools is as essential as sharing the discoveries themselves. We are active participants in, and contributors to, the Generic Model Organism Database (GMOD)

project, which seeks to bring together open-source applications and utilities that are useful to the developers of biological and genomic databases. We are contributing the software we have developed during this project to GMOD. Conversely, we reuse the Perl-based software, GBrowse, from GMOD for the visual display of our annotations.

Automated pipelines and the management of downstream data require a significant investment in software engineering. The pipeline software, the database, and the annotation tool, Apollo, as a group, provide a core set of utilities to any genome effort that shares our annotation strategy. Exactly how portable they are remains to be seen, as there is a trade-off between customization and ease of use. We will only know the extent to which we were successful when other groups try to reuse and extend these software tools. Nevertheless, the wealth of experience we gained, as well as the tools we developed in the process of reannotating the *Drosophila* genome, will be a valuable resource to any group wishing to undertake a similar exercise.

Acknowledgements

This work was supported by NIH grant HG00750 to G.M.R., by NIH Grant HG00739 to FlyBase (W.M. Gelbart), and by the Howard Hughes Medical Institute. We are grateful to our external contributors for finding the time and resources to provide additional computation pipeline results for us to consider: Karl Sirotkin (NCBI), Mark Yandell and Doug Rusch (then at Celera Genomics and now with the BDGP and TCAG respectively), and Emmanuel Mongin (Ensembl group). We also are deeply grateful to our colleague Chihiro Yamada for his valuable comments on this paper, and to Eleanor Whitfield at SWISS-PROT for providing the SWISS-PROTREAL dataset.

References

1. **Ensembl Analysis Pipeline** [http://www.ensembl.org/Docs/wiki/html/EnsemblDocs/Pipeline.html]
2. **NCBI genome sequence and annotation process** [http://www.ncbi.nlm.nih.gov/genome/guide/build.html#annot]
3. Kent JW, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM, Haussler D: **The Human Genome Browser at UCSC.** *Genome Res* 2002, **12:**996-1006.
4. Dowell RD, Jokerst RM, Day A, Eddy SR, Stein L: **The Distributed Annotation System.** *BMC Bioinformatics* 2001, **2:**7.
5. ***Saccharomyces* genome database** [http://genome-www.stanford.edu/Saccharomyces/].
6. Durbin R, Thierry-Mieg J: **A *C. elegans* database.** 1991 [ftp://rtfm.mit.edu/pub/usenet/news.answers/acedb-faq]
7. FlyBase Consortium: **The FlyBase database of the *Drosophila* genome projects and community literature.** *Nucleic Acids Res* 2002, **30:**106-108.
8. Misra S, Crosby MA, Mungall CJ, Matthews BB, Campbell KS, Hradecky P, Huang Y, Kaminker JS, Millburn GH, Prochnik SE, *et al.*: **Annotation of the *Drosophila melanogaster* euchromatic genome: a systematic review.** *Genome Biol* 2002, **3:**research0083.1-0083.22.
9. Lewis SE, Searle SMJ, Harris NL, Gibson M, Iyer VR, Richter J, Wiel C, Bayraktaroglu L, Birney E, Crosby MA, *et al.*: **Apollo: A sequence annotation editor.** *Genome Biol* 2002, **3:**research0082.1-0082.14.
10. Celniker SE, Wheeler DA, Kronmiller B, Carlson JW, Halpern A, Patel S, Adams M, Champe M, Dugan SP, Frise E, *et al.*: **Finishing a whole-genome shotgun: Release 3 of the *Drosophila* euchromatic genome sequence.** *Genome Biol* 2002, **3:**research0079.1-0079.14.
11. Benson DA, Boguski MS, Lipman DJ, Ostell J, Ouellette BF: **GenBank.** *Nucleic Acids Res* 1998, **26:**1-7.
12. Stoesser G, Sterk P, Tuli MA, Stoehr PJ, Cameron GN: **The EMBL nucleotide sequence database.** *Nucleic Acids Res* 1997, **25:**7-14.
13. Tateno Y, Imanishi T, Miyazaki S, Fukami-Kobayashi K, Saitou N, Sugawara H, Gojobori T: **DNA Data Bank of Japan (DDBJ) for genome-scale research in life science.** *Nucleic Acids Res* 2002, **30:**27-30.
14. **MySQL** [http://www.mysql.com/]
15. Date CJ: *An Introduction to Database Systems*. Reading, MA: Addison-Wesley; 1983.
16. **FlyBase GadFly genome annotation database** [http://www.fruitfly.org/cgi-bin/annot/query]
17. **Extensible markup language (XML)** [http://www.w3.org/XML/]
18. Florea L, Hartzell G, Zhang Z, Rubin GM, Miller W: **A computer program for aligning a cDNA sequence with a genomic DNA sequence.** *Genome Res* 1998, **8:**967-974.
19. Haas BJ, Volfovsky N, Town CD, Troukhan M, Alexandrov N, Feldmann KA, Flavell RB, White O, Salzberg SL: **Full-length messenger RNA sequences greatly improve genome annotation.** *Genome Biol* 2002, **3:**research0029.1-0029.12
20. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: **Basic local alignment search tool.** *J Mol Biol* 1990, **215:**403-410.
21. **WU-BLAST 2.0mp** [http://blast.wustl.edu/]
22. Adams MD, Celniker SE, Holt RA, Evans CA, Gocayne JD, Amanatides PG, Scherer SE, Li PW, Hoskins RA, Galle RF: **The genome sequence of *Drosophila melanogaster*.** *Science* 2000, **287:**2185-2195.
23. Stapleton M, Carlson J, Brokstein P, Yu C, Champe M, George R, Guarin H, Kronmiller B, Pacleb J, Park S, *et al.*: **A *Drosophila* full-length cDNA resource.** *Genome Biol* 2002, **3:**research0080.1-0080.8.
24. **The Beowulf Project** [http://www.beowulf.org/]
25. **OpenPBS Public Home** [http://www-unix.mcs.anl.gov/openpbs/]
26. **Linux networX** [http://www.linuxnetworx.com]
27. Chervitz SA, Fuellen G, Dagdigian C, Brenner SE, Birney E, Korf I: **Bioperl: standard Perl modules for bioinformatics.** *Objects in Bioinformatics Conference*, 1998 [http://www.bitsjournal.com/bioperl.html]
28. Stajich JE, Block D, Boulez K, Brenner SE, Chervitz SA, Dagdigian C, Fuellen G, Gilbert JGR, Korf I, Lapp H, *et al.*: **The Bioperl toolkit: Perl modules for the life sciences.** *Genome Res* 2002, **12:**1611-1618.
29. **bioperl.org** [http://bioperl.org/]
30. The Gene Ontology Consortium: **Gene Ontology: tool for the unification of biology.** *Nature Genet* 2000, **25:**25-29.
31. Bairoch A, Apweiler R: **The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000.** *Nucleic Acids Res* 2000, **28:**45-48.
32. Zdobnov EM, Apweiler R: **InterProScan - an integration platform for the signature-recognition methods in InterPro.** *Bioinformatics* 2001, **17:**847-848.
33. Preneel B: *Analysis and design of cryptographic hash functions.* PhD Thesis, Katholieke University, Leuven, 1993.
34. Higgins D, Thompson J, Gibson T, Thompson JD, Higgins DG, Gibson TJ: **CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice.** *Nucleic Acids Res* 1994, **22:**4673-4680.
35. **EMBOSS: showalign** [http://www.hgmp.mrc.ac.uk/Software/EMBOSS/Apps/showalign.html]
36. De Gregorio E, Spellman PT, Rubin GM, Lemaitre B: **Genome-wide analysis of the *Drosophila* immune response by using oligonucleotide microarrays.** *Proc Natl Acad Sci USA* 2001, **98:**12590-12595.
37. Stein LD, Mungall CJ, Shu S-Q, Caudy M, Mangone M, Day A, Nickerson E, Stajich J, Harris TW, Arva A, Lewis S: **The generic genome browser: a building block for a model organism system database.** *Genome Res* 2002, **12:**1599-1610.
38. **Generic Model Organism Database Construction Set** [http://gmod.sourceforge.net]
39. Mongelard F, Labrador M, Baxter EM, Gerasimova TI, Corces VG: **Trans-splicing as a novel mechanism to explain interallelic complementation in *Drosophila*.** *Genetics* 2002, **160:**1481-1487.
40. Hoskins RA, Smith CD, Carlson JW, Carvalho AB, Halpern A, Kaminker JS, Kennedy C, Mungall CJ, Sullivan BA, Sutton GG, *et al.*: **Heterochromatic sequences in a *Drosophila* whole-genome shotgun assembly.** *Genome Biol* 2002, **3:**research0085.1-0085.16.

41. **RepeatMasker documentation**
[http://ftp.genome.washington.edu/RM/RepeatMasker.html]
42. Reese MG, Kulp D, Tammana H, Haussler D: **Genie - gene finding in *Drosophila melanogaster*.** *Genome Res* 2000, **10:**529-538.
43. Burge C, Karlin S: **Prediction of complete gene structures in human genomic DNA.** *J Mol Biol* 1997, **268:**78-94.
44. Lowe TM, Eddy SR: **tRNAscan-se: a program for improved detection of transfer RNA genes in genomic sequence.** *Nucleic Acids Res* 1997, **25:**955-964.
45. Ohler U, Liao G-C, Niemann H, Rubin GM: **Computational analysis of core promoters in the *Drosophila* genome.** *Genome Biol* 2002, **3:**research0087.1-0087.12.
46. Kaminker JS, Bergman C, Kronmiller B, Carlson J, Svirskas R, Patel S, Frise E, Wheeler DL, Lewis SE, Rubin GM, *et al.*: **The transposable elements of the *Drosophila melanogaster* euchromatin - a genomics perspective.** *Genome Biol* 2002, **3:**research0084.1-0084.20.
47. ***Mus musculus* UniGene**
[http://www.ncbi.nlm.nih.gov/UniGene/query.cgi?ORG=Mm]
48. **Expressed Sequence Tags database (dbEST)**
[http://www.ncbi.nlm.nih.gov/dbEST]