Research Article:

Evolution at the Sub-gene Level: Domain Rearrangements in the
*Drosophila* Phylogeny

Yi-Chieh Wu¹,*; Matthew D. Rasmussen¹; Manolis Kellis¹,2,*

1. Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
2. Broad Institute of MIT and Harvard, Cambridge, Massachusetts, USA

*Corresponding authors:*
Yi-Chieh Wu
32 Vassar St. D510
Cambridge, Massachusetts 02139
Phone: 617-253-3434
Fax: 617-452-5034
E-mail: yjw@mit.edu

Manolis Kellis
32 Vassar St. D530
Cambridge, Massachusetts 02139
Phone: 617-253-2419
Fax: 617-452-5034
E-mail: manoli@mit.edu

**Keywords:** phylogenetics, gene fusion and fission, domain and architecture evolution

**Running title:** Sub-gene Evolution in *Drosophila*
Abstract

Although the possibility of gene evolution by domain rearrangements has long been appreciated, current methods for reconstructing and systematically analyzing gene family evolution are limited to events such as duplication, loss, and sometimes, horizontal transfer. However, within the *Drosophila* clade, we find domain rearrangements occur in 35.9% of gene families, and thus, any comprehensive study of gene evolution in these species will need to account for such events. Here, we present a new computational model and algorithm for reconstructing gene evolution at the domain level. We develop a method for detecting homologous domains between genes, and present a phylogenetic algorithm for reconstructing maximum parsimony evolutionary histories that include domain generation, duplication, loss, merge (fusion), and split (fission) events. Using this method, we find that genes involved in fusion and fission are enriched in signaling and development, suggesting that domain rearrangements and reuse may be crucial in these processes. We also find that fusion is more abundant than fission and that fusion and fission events occur predominantly alongside duplication, with 92.5% and 34.3% of fusion and fission events retaining ancestral architectures in the duplicated copies. We provide a catalog of ~9000 genes that undergo domain rearrangement across nine sequenced species, along with possible mechanisms for their formation. These results dramatically expand on evolution at the sub-gene level and offer several insights into how new genes and functions arise between species.

Introduction

Evolution can change the structure and function of genes in many ways. For example, gene duplication has long been identified as a major mechanism for generating new genes and functions (Ohno 1970, Lynch and Conery 2000, Long et al. 2003), while gene loss plays a similarly important role in shaping genomic content (Hahn et al. 2007a, Niimura and Nei 2007). These events, as well as several others such as horizontal gene transfer, gene conversion, and domain rearrangement, interact together to generate *gene families*, clusters of orthologous and paralogous genes with detectable common ancestry. By studying the genetic sequences of a family, one can infer many of the evolutionary events likely responsible for its creation.

The history of a gene family is often represented by two trees: the *gene tree*, which describes the evolutionary relationship of the genes, and the *species tree*, which describes the relationship of the species. The gene tree can be thought of as evolving “inside” of the species tree (Figure 1B). In the simplest case, these two trees are congruent (share the same topology), indicating that all the genes of the family are orthologs. However, if the two trees differ, then events such as gene duplication and loss have occurred. One can infer these events by combining several computational methods. Phylogenetic methods, such maximum likeli-
hood (Felsenstein 1981) or neighbor joining (Saitou and Nei 1987), can be used to reconstruct a gene tree and species tree from molecular sequences, and special algorithms called *reconciliation methods* (Page 1994, Goodman et al. 1979, Chen et al. 2000) can be used to determine how the gene tree fits inside, or rather *reconciles*, to the species tree. Lastly, it is the reconciliation that indicates the particular number and order of evolutionary events that have occurred in the gene family.

With the growing availability of genome sequences, this phylogenetic analysis can be applied across both sizable clades and whole genomes, in a research field called *phylogenomics* (Eisen 1998, Eisen and Fraser 2003). Many computational methods have been developed for detecting and reconstructing gene families as well as their events (Lynch and Conery 2000, Zmasek and Eddy 2002, Hahn et al. 2005, Rasmussen and Kellis 2007, Wapinski et al. 2007, Butler et al. 2009, Vilella et al. 2009, Arvestad et al. 2009, Rasmussen and Kellis 2010). This has led to a better understanding of how evolution shapes the gene content of many different species such as prokaryotes (David and Alm 2011), yeasts (Wapinski et al. 2007, Butler et al. 2009), flies (Hahn et al. 2007a), and vertebrates (Vilella et al. 2009).

Despite the sophisticated underlying models in these methods, a common assumption is to consider a gene as evolving as a single unit. However, duplications, losses, and other events can occur at the sub-gene level, and it has been suggested that homology inference be applied to domains rather than proteins (Ponting and Russell 2002). Additionally, events such gene fusion and fission challenge the current definition of a gene family, as they can form genes that have varying phylogeny and homology across the gene sequence. These more complicated events could play very important roles in generating novel genes and functions, as they are the primary source of new domain architectures that are thought to be a main source of biological complexity in the human genome and other species (Pasek et al. 2006, Yanai et al. 2002).

There are already several experimentally discovered examples of fusion and fission events. For example, *jingwei* is a chimeric gene found in *Drosophila yakuba* that arose through the fusion of the two genes *yande* (involved in nuclear mRNA splicing) and *Adh* (*alcohol dehydrogenase*). Although a fusion of genes is likely deleterious, several factors in this case have contributed to *jingwei*'s retention. First, the ancestral functions involved in this fusion event were kept intact, as *yande* is itself a recent duplicate of *yellow-emperor* and the *Adh* portion of *jingwei* is a retrotransposed copy of *Adh* (Long and Langley 1993, Long et al. 1999, Wang et al. 2000). This allowed the *jingwei* to acquire a novel function in more specific binding for long chain alcohols (Shih and Jones 2008). Second, *jingwei* has inherited the promoter sequence of *yande*, preventing degeneration of the retrotransposed *Adh* into a pseudogene. Other examples of gene fusion events in *Drosophila* gave rise to *Adh-Twain* (Jones et al. 2005), *Adh-Finnegan* (Jones and Begun 2005), *siren* (Shih and Jones 2008), *sphinx* (Wang et al. 2002), and *Quetzalcoatl* (Rogers et al. 2010), which have diverse functions in metabolic processes and male courtship behavior. In addition, studies have identified fusion and
fission events within clades such as bacteria (Suhre and Claverie 2004, Pasek et al. 2006) and fungi (Durrens et al. 2008), and specific chimeric genes have been studied in humans (Thomson et al. 2000, Courseaux and Nahon 2001) and plants (Wang et al. 2006). However, while intron phase correlations suggest that as many as ∼19% of exons in eukaryotic genes might have been formed by exon shuffling (Long et al. 1995), large-scale methods for the systematic identification and reconstruction of domain evolution and gene fusion and fission events are still lacking.

Though they do not reconstruct the history of these events, many directed studies have analyzed domain rearrangements in search of functional or evolutionary insights (Bornberg-Bauer et al. 2005, Moore et al. 2008). Quantitative analyses have shown that fusions are more prevalent to fission (Snel et al. 2000), that the number of neighbors per domain follows a power law (Apic et al. 2001; 2003) (though this could be attributed to limited coverage (Han et al. 2005)), and that specific domain combinations are more conserved than would be expected from random domain shuffling (Apic et al. 2003). Also, sequence similarity networks have been used to determine gene families of multi-domain proteins (Enright et al. 2002, Song et al. 2008, Uchiyama 2006), mechanisms of domain deletions, shufflings, and substitutions have been proposed (Weiner and Bornberg-Bauer 2006, Weiner et al. 2006), and protein interaction maps have been generated based on gene fusions (Enright et al. 1999, Enright and Ouzounis 2001).

More recently, phylogenomic methods have been developed to handle gene fusion and fission events or domain evolution, with initial approaches discovering domains de novo through sequence similarity (Snel et al. 2000), and later methods shifting to rely on underlying domain models using databases such as InterPro (Hunter et al. 2009), Pfam (Bateman et al. 2002), SCOP (Murzin et al. 1995), SMART (Schultz et al. 1998), and CDD (Marchler-Bauer et al. 2005). These studies focused on widely divergent species spanning all three domains of life and make three types of simplifying assumptions: (1) only the presence or absence of architectures in complete genomes are considered, with both architecture count and sequence information ignored (Gough 2005, Kummerfeld and Teichmann 2005, Fong et al. 2007), (2) copy numbers for architectures are considered but domain ordering is ignored, and the models have leaned towards theoretic formulations and only been applied to a limited amount of biological data (Przytycka et al. 2006, Behzadi and Vingron 2006, Wiedenhoeft et al. 2011), and (3) domain level events are mapped onto existing gene trees, with agreement between evolutionary events considered only after the independent mappings (Forslund et al. 2008).

Our work continues along these recent methods in extending phylogenomics from genes to sub-gene domains. We present the first phylogenomic approach that combines de novo discovery of sub-gene evolutionary units (which we term modules), a general model of gene evolution that captures module gain, loss, duplication, and rearrangement, and a phylogenetic reconstruction algorithm that simultaneously traces the
history of all modules while taking into account a common species tree topology. By focusing on modules, we are in many ways looking at how new genes are generated. That is, we can consider gene generation at a very low-level through mutations and insertion/deletions, or at a very high-level through gene duplication and loss. This work proposes a middle perspective that looks at gene generation through the generation of new modules and the duplication, loss, and rearrangement of existing modules.

This paper presents three distinct contributions to sub-gene phylogenomics:

- We present a method for identifying homologous modules for a family of closely related species. Our approach uses sequence similarity to define modules as the basic unit of inheritance, and therefore is not limited to existing domain databases which may be biased towards domains with known structures or domains found in well-studied proteins. We show that the resulting modules are biologically meaningful; in particular, they are frequently produced through exon shuffling events, and, when such annotations are available, they tend to keep functional domains as a single unit.

- We develop a model for gene evolution that captures architecture rearrangements, which we define as module generation, duplication, loss, merge (fusion), and split (fission) events (Figure 1D). In contrast to many previous phylogenetic approaches, our model traces gene evolution rather than architecture evolution, allowing us to explicitly capture module duplications and parallel merges and splits.

- We present a maximum parsimony algorithm STAR-MP (Species Tree informed Architecture Reconstruction - Maximum Parsimony) for inferring module architecture evolution based on (reconstructed) module phylogenies, extant module architectures, and a known species tree. Along with our evolutionary model, this algorithm is less restrictive than previous phylogenetic approaches, retaining the advantages of each. In particular, we assume a known species tree, as the added information can improve gene tree reconstruction; we do not rely on a reference gene or domain but instead view modules as the primary unit of genes, allowing us to trace the evolutionary history of genes related through any subsequence within a single reconstruction; we incorporate sequence information for each module captured through phylogenetic reconstruction; and we consider the statistical support of our reconstructions through bootstrapping. The STAR-MP software is available for download at http://compbio.mit.edu/starmp/.

To demonstrate the sensitivity and robustness of our methods, we consider eukaryotic species that are evolutionary closely related, where a species tree is well-supported and horizontal gene transfer is unlikely and not modeled. We also consider the problem of detecting architecture rearrangements at a smaller timescale, identifying only merge and split events that have occurred in recent history; we focus our analysis on the
Drosophila clade (Figure 2), as it has a dense phylogeny, a relatively recent (~60 million year old) history (Hahn et al. 2007b), and includes both close and distant species. Furthermore, at least 47 putative chimeric genes have been identified within D. melanogaster (Zhou et al. 2008, Rogers et al. 2009), and it has been estimated that ~30% of the new genes in the D. melanogaster species subgroup are chimeric (Zhou et al. 2008). We have used our methods to trace the complete history of all genes through their modules in nine Drosophila species and report numerous striking examples of architecture evolution that cannot be captured by traditional gene-level methods.

Methods

Genomic sequences and species phylogeny

Analysis was performed on nine species within the Drosophila genus: D. melanogaster, D. yakuba, D. erecta, D. ananassae, D. pseudoobscura, D. willistoni, D. mojavensis, D. virilis, and D. grimshawi. Sequences were obtained from FlyBase (May 2009 release), and we analyzed the longest protein sequence per gene and assumed a known species tree (Tamura et al. 2004) (Figure 2).

Definitions

Due to fusion and fission, a gene may contain specific domains (or more generally DNA segments) whose evolutionary history differs from the rest of the gene. Therefore, we introduce several new concepts to describe the possible relationships between such genes. Our primary unit of evolution is the module, which is a gene subsequence inherited as a single unit without internal rearrangements or breaks across the species under comparison. Modules discovered from sequence similarity are distinct from structural or functional domains of a protein, though, as we will show, they often agree. Each gene may contain one or more non-overlapping modules. These modules may share homology with other modules present within the same gene or in other genes. We call a cluster of homologous modules a module family, defined as the set of modules that descend from a single ancestral module in the last common ancestor (LCA) of all species under consideration.

For each gene, we define its architecture as the ordered list of modules it contains. Each species contains a set of genes, which corresponds to a multiset of architectures. We generalize the concept of a gene family to that of a (gene) architecture family, which contains the maximal set of genes connected by module homology. While the evolutionary histories of gene families are represented by gene trees, the histories of architecture families are represented by architecture DAGs (directed acyclic graphs), which extend gene trees by capturing module generation, fusion, and fission events, in addition to module duplication and
loss. Lastly, we define an architecture scenario as the multiset of ancestral architectures and evolutionary events mapped onto a known species tree, where each species tree node shows the type and copy number of architectures it contains, and each species tree branch shows the events that have occurred along that branch. In reconstructing architecture scenarios, we will assume a known species tree and infer ancestral architectures and events without requiring a reconciliation mapping. All trees within this work are rooted phylogenetic trees in which the leaf nodes represent extant evolutionary objects (e.g. extant species or modules in extant species) and the internal nodes represent ancestral objects (e.g. ancestral species or ancestral modules in ancestral species).

Our model for architecture evolution allows for the following evolutionary events: generation, in which a new module is created; duplication, in which an existing module is duplicated; loss, in which an existing module is lost; merge, in which two modules that appeared at the ends of two separate architectures are joined as neighbors in a single gene; and split, in which two modules that appeared as neighbors in a single gene are split and appear at the ends of two separate genes. We also make the further assumption that a module can be generated at most once. This is similar to the assumption used in Dollo parsimony in which a single generation in the last common ancestor followed by (multiple) losses is more likely than multiple independent generation events. We represent an architecture as a DAG capturing module ordering relationships between consecutive modules. Each evolutionary event corresponds to a simple graph operation (Figure 1E), and determining architecture rearrangements becomes a matter of graph rearrangements using these operations (Supplemental Section 1).

Architecture-aware phylogenomic pipeline

We present a novel phylogenomic pipeline for the architecture-aware reconstruction of gene evolution (Figure 3). The pipeline has three main stages: (1) identifying modules and module families from the genomic sequences, (2) clustering architectures into architecture families, and (3) reconstructing architecture scenarios from the architecture families and the known species tree.

Identifying modules and module families

To identify modules and their boundaries, we ran pairwise all-vs-all BLASTp comparisons (Altschul et al. 1997) between the species’ proteomes, discarding any BLAST hit with e-value > $1 \times 10^{-5}$ or percent identity < 60%. The remaining alignments were extended using LALIGN (Huang and Miller 1991), and the best hit between each query and subject pair was retained. These were re-filtered by e-value and percent identity, and short alignments (< 50 aa) and promiscuous hits (genes with > 80 hits) were removed. A list of potential
module boundaries was then found using the residue correlation matrix as in the ADDA algorithm (Heger and Holm 2003) (resolution = 10 aa, minimum module length = 30 aa), and boundaries within 30 aa of a LALIGN alignment boundary were retained. The resulting module instances were clustered into module families through OrthoMCL with default parameters (Enright et al. 2002), where the nodes represent module instances and edges are weighted by the bitscore of the LALIGN hit multiplied by the relative overlap of the modules.

Note that if desired, these steps can be replaced by matching gene sequences against a database of known structural or functional protein domains to simultaneously detect the domain boundaries and domain families. However, our approach is more general as it defines modules as evolutionarily-conserved units without relying on previous annotations. Thus, we can trace the evolutionary history of clade-specific modules, or modules that are not found in current databases (Supplemental Section 3).

As our goal is to study evolutionary events such as gene merge and split events between multiple species, we excluded any module families that appear in only a single species. Also, as in other works (Fong et al. 2007, Forslund et al. 2008), to mitigate the effects of short length repeat domains and allow for a more efficient algorithm, we collapsed tandem duplicated modules to a single copy and required that a module family appear at most once within an architecture.

Clustering architectures into architecture families

To determine architecture families, we constructed a module adjacency graph, where each vertex represented a module family, and edges were added between two modules if instances of the modules were neighbors within at least one gene. For each connected component within this graph, we identified the set of genes containing at least one module from the cluster and marked them as an architecture family.

From the module adjacency graph, we discovered several highly promiscuous module families that occur in diverse sets of genes. These module families can complicate analysis by creating very large architecture families composed of many distinct gene clusters that share little in common aside from the promiscuous module family. Therefore, we choose to analyze promiscuous module families in a separate analysis (Supplemental Section 4) and excluded them from our reconstructions. Specifically, module families were removed prior to clustering if they had more than 6 neighbors; this removed < 0.21% of all modules and < 0.38% of the modules with neighbors.

In addition, to focus on gene fusions and fissions, we filtered our architecture families to those in which one species has a gene with two neighboring modules and another species is either missing one of these modules or has no gene with these modules as neighbors.
Reconstructing architecture scenarios

For each architecture family, we reconstructed its evolution by producing an architecture scenario. This is complicated by the fact that inferring architectures in ancestral species implicitly requires inferring module counts. Rather than doing these tasks simultaneously, we adopted a three-stage approach to architecture scenario reconstruction, incorporating known rates of evolutionary events where applicable (Supplemental Section 5). First, we reconstructed the generation, duplication, and loss history of each module independently of all other modules since these events occur at the module level. Then, we then used these reconstructed module phylogenies to determine ancestral module counts, and finally, we incorporated merge and split events when inferring module groupings into architectures.

In the first stage, we incorporated known rates of evolutionary events to reconstruct the phylogenies of each module family to produce module trees. This was done by taking the peptide sequences of each module family, aligning them with the MUSCLE software package (Edgar 2004), then reverse translating the result into a (codon-aligned) nucleotide alignment. Module trees were then reconstructed from each nucleotide alignment using the SPIMAP program (Rasmussen and Kellis 2010) configured with model parameters previously determined for the *Drosophila* clade (Rasmussen and Kellis 2010), 100 pre-screen iterations, and 50 iterations.

In the second stage, we split modules trees into subtrees containing only descendants of a single common ancestor within or after the root of the species tree (i.e. proper module families). This was achieved by reconciling each module tree to the species tree using maximum parsimonious reconciliation (MPR) (Page 1994, Zmasek and Eddy 2001) and then removing any duplication nodes predating the species tree root (pre-root duplications). Each resulting subtree was then rerooted and reconciled repeatedly using MPR until all pre-root duplications were removed.

In the third stage, we reconstructed architecture scenarios for each architecture family by combining all of its module trees. From the previous steps of the pipeline, we can infer the extant architectures present at the leaves of the species tree, and we can use the reconciled module trees to infer the ancestral module copy numbers. What remains to be reconstructed is how the ancestral modules combine to form ancestral architectures and what events are responsible for their evolution.

We achieved this reconstruction using a novel maximum parsimony method called STAR-MP (Supplemental Section 2, Supplemental Figure S1), which determines the series of events (generation, duplication, loss, merge, and split) with the least total cost that explain the evolution of the given extant architectures. In this work, we used equal costs for each event, therefore minimizing the total number of events in the reconstruction. Analysis of a subset of families showed that reconstructions are robust to these costs.
STAR-MP is a dynamic programming algorithm that first works recursively up the tree to determine the cost of assigning architectures at each node, then works recursively down the tree to assign the most parsimonious architecture at each node as well as the responsible events. In the forward phase, we performed a post-order traversal of the species tree, generating a set of possible architectures for each node by finding all partitions of the available modules, then pruning the resulting list heuristically. For each possible architecture generated, we determined the operations (generation, duplication, loss, merge, split) necessary to transform it into architectures present at the child nodes. Dynamic programming was then used to find the minimum cost-to-go (e.g. minimum total cost along all descendant branches) of assigning the parent architecture. This was repeated until the root of the species tree was reached, at which point the minimum-cost architecture was assigned to the root. In the backward phase, we backtracked down the tree to determine the most parsimonious architectures and events at all the internal nodes and edges, respectively. As the maximum parsimonious reconstruction may not be unique, ties were broken randomly to arrive at a single reconstruction.

To measure uncertainty in our reconstructions, we implemented a bootstrapping procedure for STAR-MP. Each module family had 100 module trees reconstructed using SPIMAP on 100 resampled nucleotide alignments. From this set, modules trees were sampled with replacement to be reconciled and analyzed by STAR-MP 100 times, thus generating 100 bootstrapped architecture scenarios.

Validation

Input validation

A significant challenge of reconstructing architecture evolution is dealing with errors in extant genomes, e.g. resulting from sequencing, assembly, or gene model prediction. For example, erroneously connected exons in a gene model or failure to collapse multiple genes into a single gene may cause homologous modules to appear as a single gene in some species but as multiple genes in others. To validate our sequence input, we searched for errors due to gene model or assembly problems. In this section, we provide error rates based on sequence comparison or external evidence; later, in our analysis of architecture scenarios, we will show that these errors have little effect on our biological findings.

In an assembly error, a gene may be separated into multiple scaffolds, or duplicate copies of genes may appear due to undercollapsed scaffolds. In the former case, we would expect a large number of fusion/fission genes to be at the ends of scaffolds. We found 36.2% (1486) of the merge/split families to have at least one gene at the end of its scaffold; however, this large percentage is partly attributable to the presence of several...
short scaffolds in the sequenced genomes. As an alternative measure, 6.51% (2947) of genes in merge/split families are at the ends of scaffolds compared to 4.85% (6592) overall, meaning that we possibly find inflated counts for the number of merges and splits. In the latter case, we would expect near 100% identity in the sequences. Analysis of the sequences using gene spans with 2000 base pairs added upstream and downstream reveals 7.31% (300) of the merge/split families have possibly undercollapsed scaffolds (scaffolds contain undercollapsed genes with $\geq 98\%$ identity, Supplemental Section 7.1). Using our rearrangement model, we believe that such families mainly result in double-counting of duplications and losses, with little to no effect on the number of merges or splits.

To check for errors due to faulty gene models, we looked at expressed sequence tag (EST) and mRNA-seq evidence for all pairs of neighboring genes (Table 1, Supplemental Section 7.2). We found that only 0.92% (0.52%) of EST (mRNA-seq) supported neighboring gene models also had an EST (mRNA-seq) spanning both neighbors, suggesting a low rate of introns misannotated as intergenic modules. The lowest intron annotation error rate was in the well annotated \emph{D. melanogaster} genome. Larger error rates (e.g. total error rate = 11.53% (EST), 6.66% (mRNA-seq)) occur if we restrict the genes to only those that appear in architecture families (Supplemental Table S1), but this is likely attributable to the low number of EST (mRNA-seq) supported neighboring gene models in this set. Finally, note that ESTs (mRNA-seqs) only allow us to find introns misannotated as intergenic modules, e.g. spurious gene breaks, not intergenic modules misannotated as introns, e.g. missed gene breaks.

**Methods validation**

Most methods within our phylogenomic pipeline (e.g. residue correlation matrix, OrthoMCL, SPIMAP) have been evaluated in their respective works (Heger and Holm 2003, Enright et al. 2002, Rasmussen and Kellis 2010). To evaluate the last step in this pipeline, our architecture scenario reconstruction algorithm STAR-MP, we simulated module evolution, where simulation parameters were inferred using the MP architecture scenarios reconstructed from real data. Note that this reliance on MP reconstructions means that our simulations underestimate the empirical (and estimated true) event rates.

We started all simulations at the root of the species tree (as was the case for >82.6% of all MP trees) and for each simulation, generated a root architecture, where the number of module families, the number of modules per module family, and the number of connected modules were simulation parameters. To determine the events along each branch, we assumed a separate geometric distribution for each event type (generation, duplication, loss, merge, split) and each branch. The number and type of events along each branch were sampled from these geometric distributions, and an event was applied uniformly among the available modules (generation/duplication/loss), edges (split), or architectures (merge), and was discarded if it was impossible...
with the given starting architecture. Despite discarding events, event rates for the simulations were similar to the input rates (< 6% error).

Using rates estimated from the reconstructed architecture scenarios in *Drosophila*, we simulated 1000 architecture scenarios and found that STAR-MP has $\geq 63.4\%$ sensitivity and $\geq 77.8\%$ precision (Figure 4). As in the actual pipeline, the ancestral counts for each module and the architectures at the extant species were provided as input to STAR-MP, accounting for the 100% precision in generation, duplication, and loss events. Evaluation at increased event rates reveals a decrease in sensitivity consistent with a conservative MP algorithm, whereas precision degrades only slightly (Supplemental Section 7.3, Supplemental Figure S2).

**Experimental validation**

We investigated transcript evidence (EST and mRNA-seq) at the event and family level, characterizing each event or scenario as **consistent** if there exists no conflicting evidence, **inconsistent** if there exists conflicting evidence, or **unknown** if there exists no evidence (Supplemental Section 7.4).

We found that 15.1%-16.0% of scenarios are consistent and 1.1%-1.2% inconsistent, and 23.2%-40.9% of merge and split events are consistent and 0.6%-1.1% inconsistent (Supplemental Table S2). While this does not conclusively prove that the merges and splits occur, it does suggest that our reconstructed scenarios and events are not a byproduct of poor gene models.

**Results**

Using our pipeline, we found 22,813 module families combining in 14,418 architecture families, with 70.4% (10,144) of these architecture families containing only a single module, and 28.5% (4107) containing a merge or split. (All module and architecture families are available online.) The large proportion of single-module families despite such a general definition of gene and module evolution is a testament to the high specificity of our approach. The 4107 “merge/split” families consist of 12,324 module families covering 45,282 genes, and involve at least one gene from 35.9% (4457/12,431) of FlyBase gene families.

Architecture scenarios were reconstructed for 3882 families (with 10,448 module families covering 39,476 genes), of which 2818 (72.6%) had unique maximum parsimony reconstructions; the remaining 5.5% of families had many module families per architecture family and/or large ancestral counts from SPIMAP and were too complicated for MP reconstruction. Mean runtime of STAR-MP was 2.37 s with no bootstrapping and 14.40 s for 100 bootstraps. Analysis of architecture scenarios (see Common trends in architecture scenarios revealed by STAR-MP reconstruction and Genome annotation errors contribute to lineage-specific events in reconstruction) considered non-bootstrapped reconstructions. Reconstructed scenarios typically
had high bootstrap support, with a majority (63.2\%) of scenarios having a single reconstruction, e.g. 100\% support on all ancestral architectures and events. Furthermore, each event count had a low standard deviation relative to its mean (< 0.035), thus demonstrating the robustness of our reconstruction methodology.

**Module boundaries are driven by selection: comparison with domains and exons**

As our method for finding modules depended solely on sequence similarity rather than relying on previously known structural or functional domain or exon boundaries, we used these two external lines of evidence to study how modules are formed.

Using the curated Pfam-A (Version 23.0) (Bateman et al. 2002) domain definitions as a reference, we found that our module detection algorithm tends to avoid over-fragmentation (Figure 5), consistent with the idea of supra-domains (Vogel et al. 2004). Furthermore, many modules and domains are also similar in size, and many module boundaries are close to domain boundaries. Note that the long tail in Figure 5a indicates possible under-fragmentation of domains, which is expected to occur as multiple consecutive domains may have evolved jointly within the ∼60 myr *Drosophila* clade and thus have been collapsed into a single module.

Comparison between modules and exons reveals similar trends (Supplemental Figure S3), with many cases of single module-single exon or single module-multiple exons, and a large percentage (33-42\%) of modules lying precisely at an exon boundary (peak at zero distance in Supplemental Figure S3a). To study this effect further, we looked at the number of exon-bordering modules (Supplemental Table S3) and at intron-phase correlations (Supplemental Table S4). We defined an exon-bordering module as a module in which both boundaries are within ±10 residues of an exon boundary. The unusually high number of exon-bordering modules (observed = 100,974; expected = 2138; fold = 47.23; \( p < 2.23 \times 10^{-308} \), \( \chi^2 \) test) indicates exon shuffling as a prominent mechanism of module rearrangement.

Exon shuffling is also supported by a high presence of symmetrical intron phases. An intron has phase zero if it falls between two codons, phase one if it falls after the first nucleotide within a codon, phase two if it falls after the second nucleotide within a codon, and a module is labeled with the phases of its flanking introns. The splice-frame rule (Patthy 1987) states that the phases of introns flanking modules tend to match, as this prevents frameshift mutations after exon shuffling events. Similar to previous analyses (Kaessmann et al. 2002, Liu and Grigoriev 2004, Lee 2009), we found that symmetrical intron phases are enriched (O = 83,394; E = 35,003; fold = 2.38; \( p < 2.23 \times 10^{-308} \), \( \chi^2 \) test) and non-symmetrical intron phases are depleted (O = 17,580; E = 65,971; fold = 0.27; \( p < 2.23 \times 10^{-308} \), \( \chi^2 \) test). Furthermore, most of the enrichment in symmetrical intron phases is due to the presence of 0-0 modules; we believe that this enrichment reflects a tendency for exons to be reshuffled at the codon level. Interestingly, though similar trends are seen...
when comparing Pfam domains and exons (Supplemental Table S5), fold enrichments and depletions are dramatically increased for modules (e.g., fold values: exon-bordering domains = 2.32, symmetrical intron phases = 1.79, non-symmetrical phases = 0.58, \( p < 2.23 \times 10^{-308} \), \( \chi^2 \) test), and we found an abundance of 0-0 modules and a lack of 1-1 modules compared to previous analyses. These discrepancies are expected, as previous works used domain definitions produced across many genomes, whereas our modules were detected using data only across the nine *Drosophila* genomes. Regardless of whether domains or modules were used, these results suggest that modules (and domains) are produced through the shuffling of exons; here, a mutational mechanism is made apparent through module (domain) detection.

An alternative explanation for the correlation between module and domain boundaries could be their common correlation with exon boundaries. Thus, we tested whether module boundaries are depleted within domains, which would suggest that modules tend to maintain domains as a unit moreso than would be expected by exon distributions. We found that 7.1% (29,096/410,463) of introns are within ±10 residues of any module boundary, whereas within domains, this percentage decreased to 3.0% (4451/146,205), supporting our expectation that module boundaries respect domain boundaries (fold = 2.33, \( p < 2.23 \times 10^{-308} \), hypergeometric test).

**Gene ontology terms associated with rapid architecture evolution reflect adaptation**

In this section, we address whether certain functions are more likely to be involved in merge and split events. After correcting for possible biases (Supplemental Section 9), we found seven GO terms to be enriched across families with merge/split events compared to families without merge/split events (\( p < 0.001 \), hypergeometric test, FDR correction, Table 2). Interestingly, all enriched GO terms are biological processes, and almost all of them are involved in development.

We hypothesize that although gene fusions and fissions are likely deleterious for most genes, in some cases they may offer an advantage in terms of adaptability. For example, a domain may be a crucial component in several signaling pathways, each of which requires the domain to interact with a different ligand. Rather than generating the same domain multiple times throughout evolution, a species can duplicate the domain and merge it with others that encode different receptors. Such adaptability may be advantageous in signaling and development (Bhattacharyya et al. 2006, Peisajovich et al. 2010), explaining the enriched GO terms in these categories.

For example, we found an architecture scenario involving the TBP (TATA-binding protein) domain, which associates with different transcription factors to initiate transcription from different RNA polymerases. TBP
consists of a highly conserved C-terminal core that binds to the TATA box and interacts with transcription factors and regulatory proteins, and a variable N-terminal module. A study of TBP genes hypothesized that the N- and C-terminal modules may have evolved independently of each other and fused together (Sumita et al. 1993). Furthermore, TBP is dependent on upstream activators for promoter specificity; however, fusing TBP to a heterologous DNA-binding domain bypasses the need for a transcriptional activation domain, and the recruitment of TBP with an upstream activation domain provides greater flexibility in promoter arrangement (Xiao et al. 1995). Metazoans may have evolved multiple TBPs to accommodate the vast increase in genes and expression during development and cellular differentiation (Rabenstein et al. 1999).

**Protein-protein interaction datasets suggest fusion and fission of functionally complementary genes**

It has been shown that modules that merge or split tend to occur in genes with related functions (Enright et al. 1999, Enright and Ouzounis 2001, Marcotte et al. 1999). This is the basis for the Rosetta Stone model for protein-protein interaction, which suggests that given a Rosetta Stone protein with architecture AB, two proteins with architectures A and B are functionally related and more likely to interact. Possible reasons this trend are that the fusion of neighboring genes allows for tighter co-regulation (Bornberg-Bauer et al. 2005), or a single function has separated into two related genes in the case of fission. Here, we determine whether this is the case within the *Drosophila* clade. If so, we may be able to propose new functional annotations for genes.

Within *D. melanogaster*, we identified 1222 gene partners, where a gene partner consists of two genes connected by a Rosetta Stone protein. That is, for each pair of genes, we defined two sets of modules: the first set contains the modules in gene 1 but not in gene 2, and the second set contains the modules in gene 2 but not in gene 1. To be called a *gene partner*, at least one pair of modules, one from each set, must be found fused in a gene in another species. After removing the GO annotations biological process, cellular component, and molecular function, we found that 138 gene partners have both genes annotated with GO terms, and of these, 114 (82.6%) share at least one GO term. By selecting random gene partners (to control for length bias, these partners were selected from the set of 208 genes that form the 1222 partners), we observed that 61.8% share a GO term on average. This suggests that genes are more likely to have related functions if they have modules that merge or split (fold = 1.34, p < 0.001), though the cause-and-effect may be the reverse.
Common trends in architecture scenarios revealed by STAR-MP reconstruction

Our architecture scenarios that involve module merges and splits cover 4107 architecture families, 12,324 module families, and 45,282 genes. However, many of these families have very simple scenarios. Most (2295, 55.9%) contain only two modules (Figure 6), and many (1007, 24.5%) contain one gene in each of the nine species. These single gene families frequently consist of distinct subtrees, one with a single module A and another with merged architecture AB, implying a single generation and merge of module B. The second most frequent scenario (767 families, 18.7%) consists of ten sequences across nine species, corresponding to one fused gene in eight species and two fragmented genes in one species. This suggests that fragmented genes (and as we will see, fused genes) may be lineage-specific, an idea we will later revisit.

Using our 3882 reconstructed architecture scenarios, we studied the distributions of each of our events (Table 3, Figure 7, Supplemental Figure S6).

For generation events, we found that most modules (8339/10,448; or 79.8%) are generated at the species tree root (Figure 7) and were therefore inferred to exist prior to the Drosophila speciation. A previous study on the origin of new genes in the D. melanogaster species subgroup found that de novo gene origination from non-coding sequences accounts for 11.9% of new genes (Zhou et al. 2008), suggesting that partial gene origination may not be rare (Long et al. 2003).

For duplication and loss events, we observed that losses occur 2.29 times more than duplications, which is consistent with previous studies at the gene level that found factors of 1.78–3.18 (Rasmussen and Kellis 2010). The large number of duplications relative to losses arises due to paraphyletic modules (modules that appear in an ancestor but do not appear in all descendants of that ancestor), which could require multiple loss events, and also due to modules trees that are incongruent with the species tree so that during reconciliation, a single ancient duplication is compensated for with multiple losses.

Lastly, for merge and split events, a comparison of their counts revealed a 0.86:1 merge-to-split ratio, which at first seemed inconsistent with previous studies suggesting that fusion occurs more often than fission by a factor of 2.6 – 5.6 (Snel et al. 2000, Kummerfeld and Teichmann 2005, Fong et al. 2007). However, one key difference in this analysis is that we measured individual events, as opposed to simply observing the presence of fused and fragmented extant genes, and we measured events over a smaller, higher resolution time scale (the 62 myr Drosophila clade vs. all three domains of life diverging over 3.5 billion years). Furthermore, other studies do not indicate how they handle complicated events such as partial gene duplication (architecture AB to architectures AB and A) and partial gene loss (e.g. architecture AB to architecture A). We considered the former example to require a split prior to duplication and the latter to require a split prior to loss, whereas other models may have allowed for the duplication and loss to occur.
without an accompanying split. Investigation of our reconstructed architecture scenarios showed that many splits are due to such partial duplications and losses; by considering only “simple” merges and splits that are unaccompanied by generation, duplication, or loss events, the merge-to-split ratio became 5.28:1, which is much more comparable to previously determine ratios.

This last observation prompted us to also analyze the co-occurrence of events. The first trend we found is that merge and split events tend to co-occur within module and architecture families. There are 1264 scenarios (32.6% of all reconstructed scenarios, 25.9% of scenarios with merge events, 22.3% of scenarios of split events) with both merge and split events. Furthermore, 2419 module families are involved in both merge and split events (42.9% of the 5645 module families that undergo a merge, 34.3% of the 7049 module families that undergo a split). This suggests that modules that undergo a merge or split event are more likely to undergo further rearrangement (compared to the 22,861 module families in Drosophila, fold = 1.39, hypergeometric test, \( p = 1.31 \times 10^{-108} \)).

Another interesting relationship is how merge and splits events co-occur with the other events (Table 4). For example, most (74.3%) merges occur between existing (non-generated) modules, and most (92.5%) retain at least one pre-merge architecture (due to a previous duplication event). This is similar to cases such as jingwei where a duplication and merge has preserved the parental gene forms. In contrast, we found that most (52.6%) split events occur with the loss of a resulting split module, and few (34.3%) retain the pre-split architecture.

**Genome annotation errors contribute to lineage-specific events in reconstruction**

We found that 57.4% of all merge events and 78.9% of all split events occur along a branch leading to an extant species (Supplemental Table S6). This could suggest that merge and split events tend to be lineage-specific, as found in previous studies of Drosophila (Zhou et al. 2008, Rogers et al. 2009), or it could be an artifact of our pipeline arising from poor gene models and architecture annotations. For example, the D. melanogaster lineage contains 9.4% of all merge events and 16.3% of all lineage-specific merge events even though its branch accounts for only 2.9% of the total branch length within the species tree and 3.7% of the total leaf branch lengths. This genome also accounts for 14.7% (446) of the 3044 fused genes for which the split form consists of two adjacent genes, compared to an average of 10.7% (295–341) in all other genomes. However, since D. melanogaster has the best annotated genome and lowest gene model error rate (Table 1), these large percentages could be explained by genes being erroneously called as separate genes in other species and correctly called as a single gene in D. melanogaster, leading to a MP reconstruction in which a single merge event has occurred along the D. melanogaster branch.
Due to such potential anomalies, we would like a rough estimate of how many architecture families could erroneously contain merge or split events. Though we have previously validated our sequence input, we also decided to consider a highly conservative set of architecture families, which we defined as families in which no genes are neighbors, no genes are at the ends of scaffolds, and no genes have transitive BLAST hits through alternatively spliced forms. This last filter removes possible spurious gene fusions and fissions in which part of the fused gene is found in an alternative transcript but not in the longest transcript.

Filtering the 4107 architecture families involving module merges or splits resulted in a set of 2506 families (61.0% of original set) with 6120 modules (49.7%) covering 21,780 genes (48.0%). This implies that up to 39.0% of the “merge/split” architecture families could be affected by genome annotation errors or alternative transcripts that were not considered. Within the conservative set, 2492 families with 6022 modules covering 21,518 genes had reconstructed architecture scenarios. Note the two-fold decrease in the number of sequences represented. This is expected as our conservative set likely discarded many true examples of gene fusion and fission; for example, all scenarios with adjacent genes merging or a gene splitting into two adjacent genes were removed, despite both of these being valid potential mechanisms.

This conservative set of families removed 54.7% of lineage-specific merges and 48.8% of lineage-specific splits. However, 49.4% of the remaining (conservative) merge events and 79.3% of the remaining split events are still lineage-specific, and the percentage of merge events in the *D. melanogaster* lineage was only reduced from 9.4% to 6.8% (percentage of lineage-specific merge events reduced from 16.3% to 13.7%) (Supplemental Table S9), suggesting that lineage-specific events are not solely a byproduct of poor gene annotations.

Considering all architecture families, the conservative filter retained 12,408 families (86.1% of original set) with 16,178 modules (70.9%) covering 84,496 genes (75.3%). Though ratios and folds changed, all results within the previous sections hold (GO enrichment: Supplemental Table S7; PPI: Supplemental Section 11; event counts: Table 3, Supplemental Tables S8–S9, Supplemental Figure S7).

**Phylogenomic pipeline recovers previously-known examples of chimeric genes**

Zhou et al. (Zhou et al. 2008) and Rogers et al. (Rogers et al. 2009) previously identified 47 unique chimeric genes in *D. melanogaster*, 21 of which were also identified by our algorithm (Supplemental Table S10), yielding a sensitivity of 44.7%. However, Zhou et al. (Zhou et al. 2008) allowed chimeric genes to arise from a single parental sequence recruiting sequences from other intronic or intergenic sequences or from repetitive elements; this resulted in 32 of their chimeric genes having a single parental gene. Such chimeric genes might not have been detected by our pipeline since a gene subsequence must have had a hit for it to propagate through our module detection algorithm, and our use of protein sequences eliminated any
possible hits to intronic or integenic sequences. By considering only chimeric genes that have two or more parental genes, our sensitivity rises to 60% (9/15). The remaining chimeric genes were not identified due to no hits found (1), no hits found satisfying the percent identity threshold (1), frameshift mutations (1), overlapping alignments (2), or under-clustering of modules into module families (1). The first two reasons are a consequence of the BLAST step in our pipeline, where we chose thresholds consistent with previous studies in phylogenomics (Rasmussen and Kellis 2007). Similarly, regarding the last reason, we chose a clustering threshold for OrthoMCL consistent with previous studies (Enright et al. 2002).

Both Zhou et al. (Zhou et al. 2008) and Rogers et al. (Rogers et al. 2009) used BLASTn to compare CDS sequences, and they used different filters, e.g. they kept only the top hits or used different alignment length and percent identity thresholds. In our pipeline, we used peptide sequences and BLASTp to compare sequences in our pipeline as peptide homology is more sensitive than nucleotide homology. However, our choice to use BLASTp also eliminated our ability to detect frameshift mutations. Investigation of nucleotide alignments suggests that frameshift mutations account for a small percentage (∼0.58%) of total alignments and would increase the number of genes participating in merge/split families by < 3.15% (Supplemental Section 12). Future investigation may incorporate these alignments into our pipeline.

Both cases of overlapping alignments had nearly full overlaps among the three sequences, indicating that the three genes were likely duplicate copies rather than two parental sequences and one chimeric child. Aside from sequence changes in the datasets that could have caused non-overlapping alignments to now appear as overlapping, remember that we also extended our alignments using LALIGN, whereas Zhou et al. (Zhou et al. 2008) and Rogers et al. (Rogers et al. 2009) used BLASTn alignments. Manual inspection of the alignments suggests that the full overlapping alignments are correct, and the two cases correspond to non-chimeric genes.

**Gene fusion and fission events reflect a small number of common mechanisms**

In this section, we consider possible mechanisms for generating new architectures that require merges and/or splits (Figure 8), show a concrete example of the mechanism, and determine how often each mechanism occurs within *Drosophila* (Supplemental Section 13, catalog of genes by mechanism available online).

The first mechanism allows neighboring genes to merge or split, which could occur by mutations that alter start and stop codons. Allowing for the duplication of genes or subsequences before merges or splits, we found that 1681 modules and 6713 genes (16.4% and 17.2% of the modules/genes participating in merge/split events) possibly undergo this mechanism. Of course, such merges and splits are also the most suspect, as they could be caused by poor gene calls. Looking to EST (mRNA-seq) evidence, we found 274 (236) of the
above genes are inconsistent with ESTs (mRNA-seqs), 5863 (4534) genes have no ESTs (mRNA-seqs), and 576 (1943) genes are consistent with ESTs (mRNA-seqs). Other more complicated mechanisms may also explain these fusions and fissions. For example, a merged gene that is found between the ancestral split genes (not necessarily as neighbors, example in Supplemental Figure S8) may be the result of large-loop mismatch repair or replication slippage (Rogers et al. 2009). We found that 32 modules and 19 genes (0.3% and 0.05%) possibly result from these mechanisms.

The second mechanism was introduced with the case of jingwei (Supplemental Figure S9), an example which is recovered by our pipeline. Here, a retrotransposed copy of a gene is inserted into another gene and exons are combined to produce a new gene (though a fusion of the transcripts followed by retrotransposition is also possible (Akiva et al. 2006)). Such an event would correspond to a duplication and merge in our algorithm, but duplications and splits are also possible if a partial retrotransposition occurs. We found that 1904 modules and 2023 genes (18.5% and 5.18% of modules/genes participating in merge/split events) potentially result from this mechanism. In comparison, previous studies found that retrotransposition accounts for 12.2% of chimeric genes in *D. melanogaster* (Zhou et al. 2008).

The third mechanism involves segmental duplication followed by differential loss and was observed in the monkey king family (Wang et al. 2004). Though we did not find this example in our dataset as the events occur in a sister group of *D. melanogaster* not included in our nine species, we found that 60 modules and 79 genes (0.6% and 0.2% of the modules/genes participating in merge/split events) result from this mechanism. An example is the evolution of the rhea family (Figure 9).

**Discussion**

We have presented a novel model of evolution that captures module-level events such as generation, duplication, loss, merge, and split, all of which lead to new module architectures, and we have also introduced a maximum parsimony algorithm STAR-MP for tracing architecture evolution and demonstrated its accuracy in simulation. Furthermore, using our architecture-aware phylogenomic pipeline on a clade of nine *Drosophila* species, we have provided the most complete picture yet of gene and module evolution in a complete genome across multiple species.

Unlike conventional gene tree reconstruction methods, our approach incorporated module architectures and was thus able to model how genes across gene families may be related, as indicated by the presence of similar modules or architectures. Also, unlike most architecture-aware phylogenomic analyses, our approach found gene modules *de novo* rather than relying on external domain models, and our reconstruction pipeline traced gene evolution while incorporating sequence information and providing statistical bootstrapping sup-
Our results revealed that merges are more prevalent than splits as reported in previous directed studies. We also showed that merge and split events tend to occur more frequently when duplications have also retained the original architectures, likely allowing new functions to be generated by the newly-formed merged or split gene while retaining the original functions of the ancestral genes. Our approach should enable the systematic study of whether gene merges and splits are enriched in alternatively-spliced genes, and how often an alternatively-spliced form carries the original architecture. We did not focus on this question here, as we only used a single splice form (the longest polypeptide) for each gene in this initial study, and because alternative-splice forms are only well-annotated in *D. melanogaster* and not across the *Drosophila* clade.

In our study, we used SPIMAP for phylogenetic reconstruction of module trees; SPIMAP is a species tree-aware program that can maximize phylogenetic accuracy for small sequences. This is especially important as phylogenetic accuracy is dependent on the length of the sequences compared, which can make sub-gene-level phylogenetic reconstruction (as in our module trees) especially error-prone in absence of a known species tree.

We used equal event costs and ignored branch lengths (both within the known species tree and the reconstructed module phylogenies) when reconstructing architecture scenarios with STAR-MP. This assumes that events are equally likely across all branches regardless of event type or branch lengths. While we could have incorporated the inferred merge-to-split ratios (as reported in *Common trends in architecture scenarios revealed by STAR-MP reconstruction*) when assigning event costs, we wished to avoid such circular dependencies. Future studies may investigate ways to estimate these rates and incorporate them in a probabilistic or weighted parsimony framework.

A major bottleneck of architecture reconstruction algorithms is the enumeration of possible architectures, which can use both the order of modules within architectures and the number of architecture instances within families; thus, the number of possible parent architectures given two children architectures can be intractably large. STAR-MP relied on heuristics to limit the set of parent architectures for increased efficiency, and using a maximum parsimony approach, it was possible to consider a large number of parent architectures since computing the rearrangement cost for each combination of parent and children architectures is relatively fast. However, future work may require a better understanding of architecture rearrangements to better sample the full architecture space. Further analysis, for example looking at how often modules change order, may provide insight into architecture arrangements and help us develop a more biologically relevant model. Similarly, we can examine whether more complicated events such as module inversion are required for accurate architecture reconstruction.

The methods presented here relied on parsimonious reconstructions of evolutionary histories, which al-
lowed us to limit the number of scenarios to consider, resulting in high speed and accuracy. A major challenge going forward is to extend these methods to propagate sequence information across all possible reconstructions, similar to existing Bayesian and maximum likelihood phylogenetic methods, which we believe could better capture the evolutionary history of architecture families. In particular, such probabilistic methods could allow for the modeling of branch lengths within an architecture DAG (rather than being limited to architecture scenarios) and thus place evolutionary events at specific timepoints within the species history. This could also allow the simultaneous modeling of both sequence and architecture evolution, rather than the current approaches of utilizing sequence to reconstruct module trees and then either using architecture to reconstruct architecture scenarios or using reconciliation to determine module insertions and deletions.

Finally, while we have only focused on the *Drosophila* clade, increasing numbers of complete genomes are becoming commonplace across vertebrates and fungi, especially in mammals and yeast species. Further analysis of such genomes using our methods can reveal many new insights into module neofunctionalization and the emergence of new gene functions through module-level events.

Acknowledgements

We thank the MIT CompBio group for helpful comments, feedback, and discussions, and modENCODE for early release of their RNA-seq data. This material is based upon work supported under a NSF Graduate Research Fellowship to Y.W. and NSF CAREER award 0644282 to M.K.

References


Figure 1. Relationship between species trees, gene trees, and architecture scenarios. (A) Gene sequences are compared across species, and a multiple sequence alignment is constructed. Due to the presence of domains or complicated evolutionary mechanisms, these alignments may have a block structure indicating similarity at the sub-gene level. (B) In conventional phylogenetics, genes that descend from a single common ancestor are clustered into a gene family, and the history of gene families are viewed through gene trees (black lines) that evolve inside a species tree (blue area). Duplication (⋆), loss (×), and speciation (colored sub-gene blocks) events are inferred through the reconciliation of gene trees to species trees. Since each gene can belong to only a single gene family, joint histories that are evident from the architecture structure cannot be captured. (C) In sub-gene phylogenetics as presented in this work, a gene family is generalized to an architecture family in order to capture the relationships between genes with shared modules. This allows the reconstruction of gene histories to be architecture-aware, with an architecture scenario depicting more complicated events such as merges (▽) and splits (not shown). By definition, architecture scenarios use a known species tree, with architectures evolving from a parent species to a child species; thus, no reconciliation is required, and speciation events are not modeled. In this example, the joint histories of the red and teal modules are determined, including their recent merge in the branch leading to species A, corresponding to the formation of chimeric gene a₂. (D) We allow for five types of evolutionary events, two (merge and split) of which are not typically captured in conventional gene phylogenetics. (E) Gene architectures are modeled using directed graphs, with nodes representing modules and edges representing neighboring modules (within the same gene). Rearrangements of these graphs correspond to evolutionary events: adding or removing nodes correspond to generation, duplication, or loss events (not shown), and adding or removing edges correspond to merge or split events.

Figure 2. Species and phylogeny of the Drosophila clade. The phylogeny of 9 Drosophila species used in our analysis, as estimated by Tamura et al. (Tamura et al. 2004).
Figure 3. Overview of our phylogenomic pipeline. At left, the pipeline is separated into three main stages and takes as input the set of all gene sequences across several species and the known species tree relating the species. (A) In the first stage, gene sequences are compared across species, module boundaries are found, and modules are clustered according to similarity, resulting in a set of homologous module families. (B) In the second stage, a module adjacency graph is constructed based on these module families, with an edge between any two module families if at least one module instance from each family are neighbors in the same gene. Connected components of this graph define the module families to be clustered into a single architecture family. Note that B uses as input the module families determined by A, but one can use domains as determined by a database search, e.g. Pfam domains, if desired. (C) In the third stage, architecture scenarios are reconstructed for each architecture family based on a three-step procedure in which the module trees are reconstructed based on multiple sequence alignments of each module family, these module trees are reconciled to determine ancestral module counts, and the module counts, extant architectures, and known species tree are used to reconstruct the ancestral architectures and ancestral events along each branch.

Figure 4. Reconstruction accuracy of STAR-MP on simulated datasets. Event inference using STAR-MP is both sensitive and precise. Error bars show performance loss due to ties in the MP reconstruction, e.g. the MP architecture scenario and the true architecture scenario have equal costs, so events may be missed or extra events may be called in the MP reconstruction.
Figure 5. Correlation of module and domain boundaries. (A) For each module, either the overlap (# aa present in both module and domain/domain length) for modules incompletely covered by domains, or the relative size (module length/domain length) for modules completely covered by domains, was found. 75.6% of modules are equal to or larger than their corresponding domains (relative size ≥ 100%), and 28.4% of modules are of similar size to their corresponding domain (overlap ≥ 75% or relative size ≤ 150%, in gray). Bin size = 10%. (B) For each module boundary, the distance to the closest domain boundary was found, where distance = module boundary − domain boundary, blue represents left module boundaries, and green represents right module boundaries. Thus, a negative distance in blue and a positive distance in green denote that the module boundary extends further than the domain boundary. Module boundaries tend to be close to domain boundaries or extend further than the closest domain boundary. Bin size = 10 aa.

Figure 6. Distribution of architecture family sizes. (A) The number of sequences per architecture family (20 families with more than 50 sequences not shown), and (B) the number of module families per architecture family (3 families with more than 20 modules not shown) are shown. Color denotes the number of species represented in the architecture family. Many families have simple evolutionary histories, e.g. have a single gene per species or contain only two interacting modules.
Figure 7. Total counts of evolutionary events inferred on the 9 Drosophila phylogeny by STAR-MP. Many evolutionary events are inferred along each branch (counts aggregated across 3882 architecture scenarios). The large number of losses is consistent with ancient duplications followed by many compensatory losses. Many merges and splits are located along leaf branches, indicating that many fusion and fission genes may be lineage-specific. Histograms of event counts are shown along each branch, and the number of modules in a species is displayed at each species node, where counts are totaled across all architecture scenarios.

Figure 8. Mechanisms for generating fused and fragmented architectures. (A) Two adjacent genes merge into a single gene, or a single gene splits into two genes. (B) A retrotransposed copy of a gene combines with exons from another gene. (C) A chromosomal segment duplicates, and alternative portions of the duplicates are lost.
Figure 9. The inferred evolutionary history of GH22519 in D. grimshawi through duplication-degeneration of rhea. (A) The MP architecture scenario. (The full MP architecture scenario is available for download.) Most species have the module 09411 and 04568 fused in a single gene rhea. However, dgri has the two modules in separate genes, with the rhea ortholog containing module 09411 and the GH22519 gene containing module 04568. The MP reconstruction infers a split along the branch leading to dgri. Note that in the full MP architecture scenario, there is a second gene with module 09411 in the (dmel,(dyak,dere)) ancestor, which is caused by the module tree (incorrectly) grouping dmel and dere together. This results in likely spurious duplication, loss, and split events being inferred within the melanogaster subgroup. (B) A genome level view shows that rhea and GH22519 in dgri are found on two scaffolds that alternately contain orthologs to the other eight genomes. (C) The inferred evolutionary history of rhea and GH22519 in dgri through segmental duplication followed by differential degeneration. Instead of losing the entire rhea gene in one of the duplicates, rhea undergoes alternative module loss, with each copy retaining one module of the original rhea gene. This results in two genes that appear fused in the other species and fragmented in dgri.
### Table 1. EST and mRNA-seq evidence in 9 *Drosophila* genomes.

<table>
<thead>
<tr>
<th>species</th>
<th># genes</th>
<th># genes w/ EST (mRNA-seq)</th>
<th># gene pairs w/ EST (mRNA-seq)</th>
<th># gene pairs w/ spanning EST (mRNA-seq)</th>
<th>error rate (%) of EST (mRNA-seq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmel</td>
<td>14,080</td>
<td>12,640 (12,073)</td>
<td>14,052</td>
<td>78 (35)</td>
<td>0.67 (0.29)</td>
</tr>
<tr>
<td>dyak</td>
<td>16,077</td>
<td>1,618</td>
<td>15,335</td>
<td>4</td>
<td>1.80</td>
</tr>
<tr>
<td>dere</td>
<td>15,044</td>
<td>4,459</td>
<td>14,780</td>
<td>13</td>
<td>0.84</td>
</tr>
<tr>
<td>dana</td>
<td>15,069</td>
<td>5,022</td>
<td>14,680</td>
<td>24</td>
<td>1.29</td>
</tr>
<tr>
<td>dpse</td>
<td>16,099</td>
<td>2,851 (13,721)</td>
<td>15,156</td>
<td>13 (70)</td>
<td>1.86 (0.58)</td>
</tr>
<tr>
<td>dwil</td>
<td>15,512</td>
<td>4,699</td>
<td>14,442</td>
<td>17</td>
<td>0.95</td>
</tr>
<tr>
<td>dmoj</td>
<td>14,594</td>
<td>4,910 (13,035)</td>
<td>14,209</td>
<td>19 (82)</td>
<td>1.00 (0.68)</td>
</tr>
<tr>
<td>dvir</td>
<td>14,491</td>
<td>5,042</td>
<td>14,216</td>
<td>23</td>
<td>1.12</td>
</tr>
<tr>
<td>dgri</td>
<td>14,982</td>
<td>5,196</td>
<td>13,794</td>
<td>18</td>
<td>0.84</td>
</tr>
<tr>
<td>total</td>
<td>135,948</td>
<td>46,437 (39,429)</td>
<td>175,882</td>
<td>262 (187)</td>
<td>0.92 (0.52)</td>
</tr>
</tbody>
</table>

*a Two adjacent genes on the same strand.

*b Number of adjacent gene pairs in which both genes have EST (mRNA-seq) evidence.

*c Number of adjacent gene pairs in which both genes have EST (mRNA-seq) evidence and there exists at least one EST (mRNA-seq) that spans both genes.

*d Number of gene pairs with spanning EST (mRNA-seq) evidence over the number of gene pairs with EST (mRNA-seq) evidence.

### Table 2. GO enrichment for genes undergoing module rearrangement.

<table>
<thead>
<tr>
<th>rank</th>
<th>GO ID</th>
<th>GO term</th>
<th>k</th>
<th>m</th>
<th>fold</th>
<th>p-value*a</th>
<th>p-value*b</th>
<th>p-value*c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GO:0009653</td>
<td>anatomical structure morphogenesis</td>
<td>426</td>
<td>1100</td>
<td>1.36</td>
<td>1.61 × 10⁻¹⁴</td>
<td>2.13 × 10⁻⁴</td>
<td>1.08 × 10⁻⁴</td>
</tr>
<tr>
<td>2</td>
<td>GO:0048731</td>
<td>system development</td>
<td>499</td>
<td>1304</td>
<td>1.34</td>
<td>8.02 × 10⁻¹⁶</td>
<td>2.34 × 10⁻⁸</td>
<td>1.36 × 10⁻⁵</td>
</tr>
<tr>
<td>3</td>
<td>GO:0048856</td>
<td>anatomical structure development</td>
<td>557</td>
<td>1465</td>
<td>1.34</td>
<td>5.44 × 10⁻¹⁷</td>
<td>8.18 × 10⁻⁹</td>
<td>5.53 × 10⁻⁶</td>
</tr>
<tr>
<td>4</td>
<td>GO:0007275</td>
<td>multicellular organismal development</td>
<td>588</td>
<td>1554</td>
<td>1.33</td>
<td>1.97 × 10⁻¹⁷</td>
<td>3.37 × 10⁻⁹</td>
<td>3.42 × 10⁻⁶</td>
</tr>
<tr>
<td>5</td>
<td>GO:0032502</td>
<td>developmental process</td>
<td>640</td>
<td>1709</td>
<td>1.32</td>
<td>7.95 × 10⁻¹⁸</td>
<td>3.03 × 10⁻⁹</td>
<td>3.42 × 10⁻⁶</td>
</tr>
<tr>
<td>6</td>
<td>GO:0002501</td>
<td>multicellular organismal process</td>
<td>711</td>
<td>1903</td>
<td>1.31</td>
<td>1.34 × 10⁻¹⁹</td>
<td>4.23 × 10⁻¹⁰</td>
<td>8.58 × 10⁻⁷</td>
</tr>
<tr>
<td>7</td>
<td>GO:0009987</td>
<td>cellular process</td>
<td>804</td>
<td>2218</td>
<td>1.27</td>
<td>3.45 × 10⁻¹⁸</td>
<td>5.56 × 10⁻⁹</td>
<td>4.51 × 10⁻⁶</td>
</tr>
</tbody>
</table>

*a Computed using the hypergeometric test, which computes the probability of obtaining at least k annotated families for a given GO term among a dataset of size n, using a reference dataset containing m such annotated families out of N families. Here, n = 4107 and N = 14,418.

*b P-values corrected for length bias.

*c P-values corrected for length bias and multiple hypothesis testing (FDR).
Table 3. Inferred evolutionary events across architecture scenarios.

<table>
<thead>
<tr>
<th>event or ratio&lt;sup&gt;a&lt;/sup&gt;</th>
<th>G</th>
<th>D</th>
<th>L</th>
<th>M</th>
<th>M&lt;sub&gt;s&lt;/sub&gt;</th>
<th>S</th>
<th>S&lt;sub&gt;s&lt;/sub&gt;</th>
<th>D/L</th>
<th>M/S</th>
<th>M&lt;sub&gt;s&lt;/sub&gt;/S&lt;sub&gt;s&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>full&lt;sup&gt;b&lt;/sup&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># events&lt;sup&gt;d&lt;/sup&gt;</td>
<td>2109</td>
<td>4302</td>
<td>9873</td>
<td>4876</td>
<td>2952</td>
<td>5659</td>
<td>559</td>
<td>1:2.29</td>
<td>0.86:1</td>
<td>5.28:1</td>
</tr>
<tr>
<td># scenarios&lt;sup&gt;e&lt;/sup&gt;</td>
<td>1520</td>
<td>1775</td>
<td>2961</td>
<td>2242</td>
<td>955</td>
<td>2880</td>
<td>257</td>
<td>1:1.67</td>
<td>0.78:1</td>
<td>3.71:1</td>
</tr>
<tr>
<td>% scenarios</td>
<td>39.2</td>
<td>45.7</td>
<td>76.3</td>
<td>57.8</td>
<td>24.6</td>
<td>74.2</td>
<td>6.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conserved&lt;sup&gt;c&lt;/sup&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># events&lt;sup&gt;d&lt;/sup&gt;</td>
<td>1279</td>
<td>1426</td>
<td>5763</td>
<td>2567</td>
<td>1509</td>
<td>2880</td>
<td>235</td>
<td>1:4.04</td>
<td>0.89:1</td>
<td>6.42:1</td>
</tr>
<tr>
<td># scenarios&lt;sup&gt;e&lt;/sup&gt;</td>
<td>1015</td>
<td>940</td>
<td>1954</td>
<td>1374</td>
<td>529</td>
<td>1747</td>
<td>81</td>
<td>1:2.08</td>
<td>0.79:1</td>
<td>6.53:1</td>
</tr>
<tr>
<td>% scenarios</td>
<td>40.7</td>
<td>37.7</td>
<td>78.4</td>
<td>55.1</td>
<td>21.2</td>
<td>70.1</td>
<td>3.3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<sup>a</sup>G: generation, D: duplication, L: loss, M: merge, S: split. M<sub>s</sub> and S<sub>s</sub> represent simple merges and splits, i.e. merges unaccompanied by generation or duplication events, and splits unaccompanied by duplication or loss events.

<sup>b</sup>Counts aggregated across all 3882 reconstructed architecture scenarios.

<sup>c</sup>Counts aggregated across a conservative set of 2506 reconstructed architecture scenarios with limited genome annotation errors.

<sup>d</sup>Total number of events across all architecture scenarios.

<sup>e</sup>Number of architecture scenarios with at least one branch having the event type.

---

Table 4. Retainment of ancestral architectures by merge and split events.

<table>
<thead>
<tr>
<th>MERGES</th>
<th>all</th>
<th>w/o generation</th>
<th>w/ generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of events&lt;sup&gt;a&lt;/sup&gt;</td>
<td>4876</td>
<td>3623 (74.3%)</td>
<td>1253 (25.7%)</td>
</tr>
<tr>
<td>retained at least one split architecture&lt;sup&gt;b,c&lt;/sup&gt;</td>
<td>4512 (92.5%)</td>
<td>3437 (94.9%)</td>
<td>1075 (85.8%)</td>
</tr>
<tr>
<td>retained both split architectures&lt;sup&gt;c,e&lt;/sup&gt;</td>
<td>2688 (55.1%)</td>
<td>2688 (74.2%)</td>
<td>n/a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SPLITS</th>
<th>all</th>
<th>w/o loss</th>
<th>w/ loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of events&lt;sup&gt;a&lt;/sup&gt;</td>
<td>5659</td>
<td>2683 (47.4%)</td>
<td>2976 (52.6%)</td>
</tr>
<tr>
<td>retained merged architecture&lt;sup&gt;d,e&lt;/sup&gt;</td>
<td>1943 (34.3%)</td>
<td>1844 (68.7%)</td>
<td>99 (3.3%)</td>
</tr>
</tbody>
</table>

<sup>a</sup>Total number of merge/split events, as well as whether these events are merges with a newly generated module (e.g. A → AB) or splits that also lose an associated split module (e.g. AB → A). Percentages out of the total number of merge/split events.

<sup>b</sup>Number of merges that retain at least one ancestral split architecture (e.g. A,B → A,AB).

<sup>c</sup>Number of merges that retain both ancestral split architectures (e.g. A,B → A,B,AB).

<sup>d</sup>Number of splits that retain the ancestral merged architecture (e.g. AB → AB,A,B).

<sup>e</sup>Percentages out of the number of events in the top row.
A **Alignment**

B **Interpretation in traditional gene phylogenies**

C **Interpretation in sub-gene phylogeny**

D **Evolutionary events**

E **Directed graph models**
From **architecture scenarios**:  
- generations/duplications/losses/merges/splits

From **modules**:  
- boundaries driven by exons and domains

From **architecture families**:  
- functional enrichment  
- functional annotation

**Identify modules and module families**

**Find sequence similarity**: (pairwise BLAST + LALIGN)

**Determine module boundaries**: (residue correlation matrix)

**Cluster module instances**: (OrthoMCL)

**Build module trees**: (SPIMAP)

**Determine ancestral module counts**: (maximum parsimony reconciliation)

**Build architecture scenarios**: (STAR-MP)

**Species tree**

A  
B  
C

**Extant architectures**

A  
B  
C

**Make clusters using connected components on module adjacency graph**

**Build module trees** (if bootstrapped, sample alignments)

**Determine ancestral module counts** (maximum parsimony reconciliation) (if bootstrapped, sample module trees)

**Build architecture scenarios** (STAR-MP)
A: Number of architecture families as a function of the number of genes per architecture family.

B: Number of architecture families as a function of the number of module families per architecture family, with different colors indicating different species (2 to 9).
A) fusion/fission of adjacent genes

B) retrotransposition and exon shuffling

C) duplication-degeneration
Figure 1: Genome evolution in D. melanogaster. (A) Phylogeny of the Drosophila species used in this study. (B) Example of genome evolution involving segmental duplication and differential degeneration. (C) Detailed view of the genome evolution at the level of individual scaffolds and modules.

Legend:
- 5' modules
- 3' modules
- Exons
- Dark = + strand
- Light = - strand

6. Segmental duplication and differential degeneration

- Ancestral genome 7
- Foil ergic53
- Dgri
- Activation of lineage-specific modules
- Removal of lost regions

Different scaffolds:
- 3L
- Scaffold 15110
- Scaffold 15072

Reference gene expression data:
- CG6673
- CG6662
- CG32352
- CG32355
- Nelf-E
- RpL14
- CG5989
- CG6282
- CG5978
- GH22519
- GH15312
Evolution at the Sub-gene Level: Domain Rearrangements in the
Drosophila Phylogeny –
Supplementary Material

Yi-Chieh Wu, Matthew D. Rasmussen, Manolis Kellis

Supplementary Methods and Results

In this supplement, we let architectures refer to both a single architecture, e.g. \( a \), \( ab \), and a multiset (unordered list) of architectures, e.g. \( \{a\} \), \( \{ab\} \), \( \{a, ab\} \), \( \{a, a, ab\} \). Architectures can therefore be assigned to both genes (using the single architecture definition) and to entire species (using the multiset of architectures definition).

1 Modeling architecture rearrangements

Consider the problem of transforming a parent architecture to a child architecture. We represent these architectures as directed acyclic graphs, where the in-degree and out-degree of a node are each at most one. A module is indicated by a node, and neighboring modules within the same gene are joined with a directed edge. The problem is now equivalent to transforming from the parent graph to the child graph, where each allowable graph operation corresponds to an evolutionary event (Figure 1c):

- **G**: Add a node (corresponding to a module that does not currently exist).
- **D**: Duplicate a node.
- **L**: Remove a node.
- **M**: Add an edge between two (existing) nodes.
- **S**: Remove an edge between two (existing) nodes.

Note that these definitions require that generation, duplication, and loss occur at the module level. For example, generating a sequence of multiple modules is only possible through generation of the component modules. This assumption treats modules as the basic building blocks of a gene and implies that the generation, duplication, and loss of larger sequences (as measured by the number of modules) incur a higher cost.

Evolutionary events are applied in the following order: generation, duplication, loss, split, merge. There are also some caveats to account for the architecture representation. For example, we allow for the duplication and loss of entire architectures or sub-architectures, where the cost of duplication/loss corresponds to the number of modules duplicated/lost. Furthermore, the duplicated (sub-)architecture retains all the edges of the original, and edges within a lost (sub-)architecture can be removed without penalty. Duplication/loss of a sub-architecture may also incur one or two (hidden) splits, depending on whether the left and right end of the sub-architecture were connected to another module in the parent architecture; this prevents parental sub-architectures from appearing in isolation in the child without penalty. Finally, merges are free between two generated modules (but not between a generated module and an existing module); this allows for the generation of architectures in addition to the generation of modules.

Note that with this representation, there is a one-to-one mapping between nodes in the parent and child graphs (using the “null” node as the parent of a generated node and the child of a lost node). Tracing this mapping for each module in a reconstructed architecture scenario reveals the series of generations,
duplications, and losses that have led to the extant module counts, and tracing the addition or removal of edges across the mappings reveals the series of merges and splits that have led to the extant architectures.

To demonstrate the robustness of our architecture model, consider the case of a duplicated sub-architecture. For example, rearranging parent architecture \( A = \{abc\} \) to form child architecture \( B = \{ab, abc\} \) would infer a duplication of module \( a \), a duplication of module \( b \), and a (hidden) split between modules \( b \) and \( c \). Note that (1) the original parent architecture \( abc \) is retained in \( B \) (with no inferred events), (2) no merge event is required between \( a \) and \( b \) since the multi-module sub-architecture \( ab \) arose from \( abc \), and (3) a (hidden) split is inferred to account for the missing edge between \( b \) and \( c \) when \( ab \) is duplicated from \( abc \).

### 2 Architecture scenario reconstruction

STAR-MP is a maximum parsimony algorithm for reconstructing the architecture scenario such that the total evolutionary cost is minimized. That is, for each architecture family, given the known species tree, the known architectures at the leaves, and the (inferred) module counts at the ancestors, STAR-MP infers the architectures at the ancestors and the events that occur along the branches. Here, we describe our architecture model and present the pseudocode for STAR-MP, which was described in brief in *Reconstructing architecture scenarios*. In the pseudocode, details may be omitted for clarity.

#### 2.1 Notation

- **\( T \)**: the rooted, full, binary species tree, with nodes \( N(T) \), edges \( E(T) \), root \( R(T) \), and leaves \( L(T) \)
- **\( n \in N \)**: a node with parent \( n_p \) and left and right children \( n_l \) and \( n_r \)
- **\( A \)**: an architecture with modules \( A^r \)
- **\( x \in \{G, D, L, M, S\} \)**: an evolutionary event with associated costs \( cost(x) \)
- **\( e = \{x_i\} \)**: a (multi)set of events
- **\( P(n) \)**: the modules at node \( n \)
- **\( Q(n) \in \{A_i\}, Q^*(n) = A \)**: the possible architectures, and the optimal (min-cost) architecture, at node \( n \)
- **\( E(A, B) = e, Cost(A, B) \in [0, \infty) \)**: the optimal set of events and the optimal cost to transform architecture \( B \) to architecture \( A \)
- **\( F(n, A) \)**: the optimal cost-to-go of assigning architecture \( A \) to node \( n \), e.g. the optimal cost of events up to node \( n \) such that \( Q^*(n) = A \)
- **\( G(n, A) \)**: the traceback pointer for \( F(n, A) \)
- **\( H(n, n_l), H(n, n_r) \)**: the optimal events along edges \( n \to n_l \) and \( n \to n_r \)

We also define the following operations on architectures: \( A + B \) is the architecture containing the architectures in \( A \) and \( B \), \( A \cup B \) is the multiset of architectures that appear in either \( A \) or \( B \), and \( A \cap B \) is the multiset of architectures that appear in both \( A \) or \( B \). For example, if \( A = \{a, ab\} \) and \( B = \{a, bc\} \), then \( A + B = \{a, a, ab, bc\} \), \( A \cup B = \{a, ab, bc\} \), and \( A \cap B = \{a\} \). Similar operations will also be used for modules.
2.2 Pseudocode

```
Input: T, cost(x) ∀x ∈ {G, D, L, M, S}, P(n) ∀n ∈ N(T), Q(n) ∀n ∈ L(T)
Output: The reconstructed architecture scenario, as defined by Q∗(n), H(n, nl), H(n, nr) ∀n ∈ N(T)

/* Initialization */
for n ∈ L(T) do
  F(n, Q(n)) = 0
end

/* Recursion */
for n ∈ N(T) (post-order traversal) such that n /∈ L(T) do
  // Determine possible architectures
  Q(n) = ∅
  for B ∈ Q(nl), C ∈ Q(nr) do
    update Q(n) with archgen(P(n), B, C)
  end
  // Determine min-cost-to-go architecture
  for A ∈ Q(n) do
    // Determine rearrangement cost
    for B ∈ Q(nl) ∪ Q(nr) do
      E(A, B), Cost(A, B) = archeventcosts(A, B, {cost})
    end
    // Find optimal
    for B ∈ Q(nl), C ∈ Q(nr) do
      J(B, C) = F(nl, B) + Cost(A, B) + F(nr, C) + Cost(A, C)
    end
    G(n, A), F(n, A) = arg min / minB∈Q(nl),C∈Q(nr) J(B, C)
  end
end

/* Termination */
Q∗(R(T)), F∗ = arg min / minA∈Q(R(T)) F(R(T), A)

/* Traceback */
for n ∈ N(T) (pre-order traversal) such that n /∈ L(T) do
  Q∗(nl), Q∗(nr) = G(n, Q∗(n))
  H(n, nl), H(n, nr) = E(Q∗(nl)), E(Q∗(nr))
end
return Q∗(n), H(n, nl), H(n, nr) ∀n ∈ N(T)
```

**Algorithm 1: STAR-MP pseudocode.** Given the species tree, the ancestral module counts, and the extant architectures, STAR-MP reconstructs the architecture scenario by inferring the ancestral architectures and the events along the branches. Note that an explicit cost cost(G) for generation is not required since modules are generated at the LCA of all species with the module. Code to account for ties is omitted for clarity; ties were broken arbitrarily in our implementation.
Algorithm 2: archgen pseudocode for generating possible architectures. Given the available modules at the parent and the children architectures, archgen finds the possible parent architectures.
**Input:** parent architecture $A$, child architecture $B$, evolutionary costs $costs$

**Output:** the optimal series of events and the optimal cost required for the rearrangement

/* Remove architectures that appear in both parent and child */
for $arch \in A \cap B$ do
    // do not remove if it is the last instance in the parent so it can be used in duplication
    if count of $arch$ in $A > 1$ then
        remove $arch$ from $A$ and from $B$
    end
end

/* Make graph representations */
$G_A, G_B = \text{directed graph representations of } A \text{ and } B$

/* Generate/Duplicate/Lose modules through GDL, then Merge/Split modules through MS */
$E = \text{empty list} // \text{series of events required for rearrangement}$
for $(E_{GDL}, G_{GDL}, gen, splits) \in \text{GDL}(G_A, G_B)$ do
    for $E_{MS} \in \text{MS}(G_{GDL}, G_B, gen, splits)$ do
        // put GDL and MS events together and store
        append $E_{GDL} + E_{MS}$ to $E$
    end
end

/* Find optimum: findcost($e, costs$) determines the total cost of events $e$ given costs $costs$ */
$E^*, C^* = \arg \min_{E \in E} \text{findcost}(E, costs)$
return $E^*, C^*$

**Algorithm 3:** archeventcosts pseudocode for finding the optimal events and cost for an architecture rearrangement. Given a parent architecture, a child architecture, and the cost of evolutionary events corresponding to elementary graph operations, archeventcosts finds the optimal series of events and the optimal cost to transform the parent architecture to the child architecture. This is split into two parts: (1) generation, duplication, loss, and possibly split of modules until the parent and child architecture have the same number of modules, and (2) merge, split of modules until the parent architecture and child architecture are equivalent.
Input: parent architecture graph $G_A$, child architecture graph $G_B$

Output: series of events, resulting graphs, generated modules, and incurred splits for the rearrangement

/* Generations - modules are generated (and then possibly duplicated) as single modules */
$E_G = \emptyset$; gen, duplst, loselst = empty dictionaries // generated module and dicts of [reg,# of dups/losses]

for reg $\in$ set of modules in $G_A$ and $G_B$
do
  $a, b =$ count of reg in $G_A$ and $G_B$
  if $a = b$ then continue; // no change in module count
  else if $a = 0$ then
    add reg to gen, add $b$ instances of reg to $G_A$, append GEN of reg and $b - 1$ DUP of reg to $E_G$
  else if $a < b$ then duplst[reg] = $b - a$; // dups
  else if $a > b$ then loselst[reg] = $a - b$; // losses
end

/* Duplications - one element for each combination of dups. Also keep track of duplication sources (map from duplicated child module to source parent module). */
$E_D, G_D, dupmap = \emptyset$ // list of events, graphs, and maps due to duplication

if duplst is empty then append copy of $E_G$, copy of $G_A$, empty map to $E_D, G_D, dupmap$; // no dups
else
  // duplicate parent architectures until there are enough modules - Afterwards, there may be extra modules due to multi-module duplications. These will be lost in the Loss phase.
  for each unique combination of architectures that are satisfied do
    $E, G, map =$ copy of $E_G$, copy of $G_A$, empty map; append $E, G, map$ to $E_D, G_D, dupmap$
    for arch $\in$ todup do add arch to $G$, add DUP of reg $(\forall reg \in arch)$ to $E$, update map
  end
end

/* Losses - one element for each combination of dups/losses */
for $E, G, dmap \in E_D, G_D, dupmap$
do
  recalculate loselst (as in Generation phase) // recalc due to presence of extra duplicated modules
  if loselst is empty then yield copy of $E_G$, copy of $G_A$, gen, $\emptyset$ and continue; // no losses
  // calculate losses for each combination of lost nodes
  for each unique combination of modules to lose s.t. the counts in loselst are satisfied do
    $E_L, G_L = \emptyset$ // remove modules
   _splits = $\emptyset$ // count a split (from source) only once
    for $u \in$ tolose do
      // Consider $t - u - v$ and $t_s - u_s - v_s$, where $n_s = dmap[n]$ (e.g. dup source) if $n$ arose from a dup and $n_s = n$ o.w.
      // For each edge, remove the edge, and incur a split if both modules are not lost and the source of the edge was not already split.
      if $\exists l$, remove $(t, u)$ from $G_L$, add $(t_s, u_s)$ to splits, append SPLIT of $(t, u)$ to $E_L$ if $(t_s, u_s) \notin$ splits
      do same for $(u, v)$ and $(u_s, v_s)$
      // remove the module
      remove $u$ from $G_L$, append LOSS of $u$ to $E_L$
    end
    // remove redundancies (duplication and loss of same node)
    for $u \in$ dmap $\cap$ tolose do remove DUP of $u$ and LOSS of $u$ from $E_L$
yield $E_L, G_L, gen, splits$
end
end

Algorithm 4: GDL pseudocode for rearranging architectures through generation/duplication/loss events. Given a parent architecture and a child architecture, GDL finds the series of generation/duplication/loss events so that the resulting parent and child architectures have the same number of modules. Note that (1) which nodes are duplicated affects resulting architectures and possibly incurs extra splits and losses, and (2) which nodes are lost affects resulting architectures and possibly incurs extra splits; thus, GDL considers each possible combination of duplications and losses.
### Algorithm 5: MS pseudocode for rearranging architectures through merge/split events.

Given a parent architecture and a child architecture, MS finds the series of merge/split events so that the resulting parent and child architectures are equivalent.

| Input: parent architecture graph $G_A$, child architecture graph $G_B$, generated modules $\text{gen}$, incurred splits $\text{splits}$ during generation/duplication/loss |
| Output: series of events required for the rearrangement |

```plaintext
/* Remove architectures that appear in both $G_A$ and $G_B$ */
for $\text{arch} \in G_A \cap G_B$ do
  remove $\text{arch}$ from $G_A$ and $G_B$
end

/* No merge/splits if no architectures left */
if $G_A$ is empty $\land$ $G_B$ is empty then
  yield empty list and stop iteration
end

/* Permute the nodes of one graph to look at all node-to-node assignments between graphs */
for pairwise assignment of nodes between $G_A$ and $G_B$ s.t. paired nodes have same module type do
  // add/remove edges (incur M/S events)
  $G_A^{\text{copy}}$ = copy of $G_A$
  $E$ = empty list
  // split (remove edge from $G_A^{\text{copy}}$) - must do first in case of insertion/deletion
  for $(u_A, v_A) \in G_A$ do
    $(u_B, v_B) = \text{assignment of } (u_a, v_a)$ from $G_A$ to $G_B$
    if $(u_B, v_B) \notin G_B$ then
      remove $(u_A, v_A)$ from $G_A^{\text{copy}}$
      // do not penalize if this split was already incurred during duplication/loss
      if $(u_A, v_A) \notin \text{splits}$ then
        $(u, v) = \text{module types corresponding to } (u_A, v_A)$
        append SPLIT of $(u, v)$ to $E$
      end
    end
  end
  // merge (add edge to $G_A^{\text{copy}}$)
  for $(u_B, v_B) \in G_B$ do
    $(u_A, v_A) = \text{assignment of } (u_B, v_B)$ from $G_B$ to $G_A$
    if $(u_A, v_A) \notin G_A$ then
      add $(u_A, v_A)$ to $G_A^{\text{copy}}$
      // do not penalize if both $u$ and $v$ were generated
      if $u \notin \text{gen} \lor v \notin \text{gen}$ then
        $(u, v) = \text{module types corresponding to } (u_B, v_B)$
        append MERGE of $(u, v)$ to $E$
      end
    end
  end
  yield $E$
end
```

3 Using domain annotations

As noted in the main text, the step of ‘identifying modules and module families’ in our phylogenomic pipeline may be replaced by a database search against existing domains (e.g., Pfam, SCOP, SMART, CDD, etc). We have chosen to use a de novo approach to module identification rather than using a domain database search for a number of reasons.

Our main reason is that we wished to make no a priori assumptions about the identity or boundaries of the modules. As mentioned in the main text, domain databases are often biased, for example, towards
domains with known structures or function. However, our definition of modules is evolutionarily-based and depends solely on sequence conservation.

An analysis of genome coverage (excluding singleton domains or modules) also revealed that only 62% of Drosophila genes have Pfam annotations compared to 82% of genes with module annotations. If we include singleton domains/modules, the change in coverage for Pfam annotations is negligible while the coverage for module annotations increases to 85%, with the remaining 15% of genes lacking BLAST hits that pass our filters. We believe that this difference in coverage is because our approach captures both known and unknown domains; in particular, it captures domains that are evolutionarily (rather structurally or functionally) conserved.

In addition, domain definitions are compiled using genomes across the three domains of life, meaning that domain families may be overclustered when looking at a small subset of genomes such as the ∼60 myr Drosophila clade. (Recall that gene and domain families are defined as the set of genes/domains that descend from a single gene/domain in the most recent common ancestor of all species under consideration. Therefore, restricting the genomes to a small subset will break the original families into many smaller clusters.) A major benefit of our approach is that it can be used at multiple timescales: we can look across the three domains of life as in domain databases, or we can find novel clade-specific domain families that may be missing from domain databases, as in our analysis of Drosophila. For comparison, ADDA found a number of novel domain families missing from Pfam and SMART, with the majority of these new families specific to a single domain of life. Such novel domain families may also be present within the Drosophila phylogeny (perhaps to confer clade-specific biological functions), and a such using domain definitions compiled across all three domains of life may lack the power to detect such recently evolved families.

Our approach can also capture known and unknown domains and neutral evolutionary events. In particular, we can identify modules linked to a protein function but associated with an unknown domain. This is important, as we are also interested in analyzing genome-wide event rates (or counts), and if we focus on the subset of genes in which the merged or split domain has a known function, these rates (counts) may be biased.

Finally, our approach has a higher power than a database search, and moreover, it can be applied to newly sequenced genomes to discover new modules. In particular, we can analyze a group of closely related genomes that are together distant from other genomes. As mentioned previously, our method will find novel domain families that have evolved solely within the newly sequenced clade without requiring these families to be defined in domain databases.

4 Promiscuous modules

Analysis in our main text (Results) excluded promiscuous modules in our pipeline. Including these in our analysis decreased the number of architecture families to 14,156 (1.8% decrease), with 4201 families containing more than one module (0.03% decrease in ratio of # families with ≥ 2 modules/# families) and 4037 families containing a fusion or fission (<0.01% decrease in ratio). These “fusion/fission” families consist of 12,567 module families (0.2% increase) covering 46,100 sequences (0.2% increase) and involve at least one gene from 4533 (36.8%) of gene families (0.6% increase). As expected, the distribution of architecture families also shifted; for example, there would be 11 families with more than 20 modules and 22 families with more than 50 sequences (compare to Figure 6). Such increases are particularly problematic for our STAR-MP architecture reconstruction algorithm and would likely result in increased runtime or be too complicated for MP reconstruction.

5 Incorporating known rates of evolutionary events

We have incorporated known tendencies in event costs where applicable in our pipeline; for example, we used known estimated duplication and loss rates in Drosophila (Hahn et al. 2007) to reconstruct module phylogenies with SPIRMAP. However, systematic studies of merge and split events (Snel et al. 2000, Kummerfeld and Teichmann 2005, Fong et al. 2007) have only determined total counts or merge-to-split ratios, and these are neither specific to the Drosophila clade nor do they incorporate architecture counts. The few studies on fusion and fission in Drosophila (Zhou et al. 2008, Rogers et al. 2009) focus on a subset of species and on
genes with significant experimental evidence. In contrast, it takes a systematic, genome-wide approach to determine event rates that are unbiased and reflective of the entire genome.

6 Cost of evolutionary events

We analyzed a subset of 200 families using varying event costs. Note that the cost of generation does not affect STAR-MP reconstructions since each module is assumed to have been generated only once at the most recent common ancestor of all species that contain the module. In addition, in our current implementation, duplication and loss costs also have limited effect since ancestral counts are inferred using the reconstructed module phylogenies, and a module that is duplicated and subsequently lost along the same branch does not incur any cost.

We tested six settings for the event costs: one in which all events were equal (as in the main text), four in which a single event (D,L,M,S) had twice the cost of the others, and one in which merges and splits were twice the cost of generations, duplications and losses. For each setting, we summed the number of inferred evolutionary events of each type, then computed its deviation from the inferred counts under equal costs. All deviations were less than 3.7% except for four cases: when merges had double the cost, the number of merges decreased by 23.9% and the number of splits increased by 20.3%, and when splits had double the cost, the number of merges increased by 36.6% and the number of splits decreased by 26.8%. Furthermore, while the number of inferred regions are the same across all settings (since we have used the same input module trees), the number of genes for a doubled merge cost decreased by 2.4% and for a doubled split cost increased by 2.9%. This is consistent with our expectations, as a higher merge cost should result in a larger number of merged ancestral genes (e.g. fewer genes given the same number of modules) so that fewer merges and more splits are inferred. Similarly, a higher split cost should result in a larger number of split ancestral genes (e.g. more genes given the same number of modules) so that more merges and fewer splits are inferred.

However, in this smaller set of families, each reconstruction contributes a larger portion to the total number of events; thus, many deviations could be attributed to the small number of families that have multiple maximum parsimonious reconstructions. (Remember that ties are broken randomly.) If we consider the 143 families for which only a single maximum parsimonious reconstruction exists for every setting, almost all deviations drop two-fold or more. For this filtered set, all deviations were less than 1.1% except for two cases: when splits have double the cost, the number of merges increased by 22.0% and the number of splits decreased by 9.1%. Note that a doubled merge cost negligibly affects the inferred evolutionary events, and that the deviations in merge and split counts for a doubled split cost have dropped. We believe that many of the deviations for the doubled split cost are due to cases in which an architecture is partially lost. Here, the high split cost causes STAR-MP to infer a split ancestral gene when other parameter settings would infer a merged ancestral form. (See also Supplemental Section 7.3.)

Our analysis shows that in almost all cases, the balance of inferred events is maintained since these events are constrained by the evolutionary evidence. Significant deviations may be seen if a higher split cost is used, but then, a larger number of merges and a lower number of splits will be inferred, which would further support our findings that merges are more prevalent than splits (Common trends in architecture scenarios revealed by STAR-MP reconstruction).

7 Validation

7.1 Detection of undercollapsed scaffolds

The BLASTp hits of a species versus itself were filtered to retain hits between genes located on different scaffolds and with percent identity ≥ 95%. These were run through LALIGN using nucleotide sequences extended to ±2000 nt upstream and downstream, and hits with percent identity ≥ 98% were retained. An architecture family is said to contain possibly undercollapsed scaffolds if at least two genes within the family have a hit in this final list.
7.2 EST and mRNA-seq evidence

ESTs were obtained from GenBank and compared against the protein sequences using BLASTx. Only hits with long ESTs (≥ 250 nt), e-value ≤ 1 × 10^{-5}, percent identity ≥ 96%, and alignment length ≥ 50 aa were retained. If a single EST aligned to the same sequence in multiple places, the alignment with the highest percent identity was retained. An EST is said to span two genes if it aligns with both genes and the alignments are in the same direction and overlap by ≤ 15 aa.

mRNA-seqs at 36 and 75 nt resolution were obtained from modENCODE (http://www.modencode.org). Briefly, this protocol used polyA RNA extracted from D. melanogaster, D. pseudoobscura, and D. mojavensis male and female heads, with sequences aligned to the genome with Bowtie allowing for up to two mismatches. We mapped the mRNA to genes based on genome location, and an mRNA-seq was said to span two genes if it aligns with both genes and the alignments are in the same direction and do not overlap.

7.3 Simulations under various event rates

Keeping the generation rate constant at 1X the estimated true rate, we set the duplication, loss, merge, and split rates at 1X, 2X, and 4X the estimated true rates. We tested five different settings, simulating 1000 architecture scenarios for each setting (Supplemental Figure S2).

In general, STAR-MP has higher precision than sensitivity for any given event, and performance tends to degrade as the event rates increase and the true architecture scenarios become more complicated. Indeed, part of the decrease in sensitivity can be attributed to trying to explain more complex architecture scenarios with a conservative MP algorithm. STAR-MP also tends to have higher generation, duplication, and loss performance than merge and split performance, and as expected, generation, duplication, and loss performance is consistent across various merge and split rates. Interestingly, merge performance is typically higher than split performance. Further investigation showed that low split performance can be attributed to cases in which an architecture is partially lost. Here, the true reconstruction is a merged parent architecture undergoing a split and loss to result in the surviving sub-architecture; however, STAR-MP tended to reconstruct a split parent architecture so that only a loss is needed to produce the surviving sub-architecture.

7.4 Support for fusion and fission events using transcript evidence

We excluded from this analysis the 6.1% (249/4107) of merge/split scenarios have no merge or split events. This occurred since we determined merge/split families based solely on the clustering of architectures into architecture families. However, all merge and split events in the family may have occurred prior to the species tree root; such a case can only be determined after the architecture scenario reconstruction.

Consider a merge or split event between two modules, as found by our architecture scenario reconstruction algorithm. Each of these events bifurcates the leaves, with one subset containing the leaves belonging in the subtree descended from the event, and the other subset consisting of the the rest of the leaves. We denoted these sets as the merged genes or split genes, respectively, if we were considering a merge event, or vice-versa if we were considering a split event. A merged gene was classified as consistent if there existed at least one EST/mRNA-seq that covers the boundary between the modules, and as unknown otherwise. (Note that we did not allow a merged gene to be inconsistent with the evidence.) A split gene was classified as inconsistent if there existed at least one spanning EST (mRNA-seq) (Supplemental Section 7.2) for the gene, as consistent if there existed an EST but no spanning EST (mRNA-seq), and as unknown otherwise. Once the genes were classified, an event was classified as consistent if there existed at least one consistent merged gene and at least one consistent split gene, as inconsistent if there existed at least one inconsistent split gene, and as unknown otherwise. Finally, the events for each scenario were pooled, and each scenario was classified as inconsistent if any event was inconsistent, as consistent if all events were consistent, and as unknown otherwise.

8 No substitution rate bias in merge/split families

Analysis of Adh-derived chimeric genes previously revealed elevated rates of amino acid substitution after merge events (Long and Langley 1993, Jones et al. 2005, Jones and Begun 2005). It was speculated that
the new function of the chimeric gene no longer required strong conservation, or that amino acids along the merge boundary rapidly evolved to repair any possible damage incurred by the merge event. To examine whether this bias occurs at a systematic level, we computed the substitution rate for each module family (Supplemental Figure S5) using the SPIMAP model of assigning a gene-specific (here module-specific) rate and species-specific rate to each tree (Rasmussen and Kellis 2007; 2010). Notice that while the distributions are significantly different, the effect size is small (fold of median rates = 0.970–0.975). Furthermore, contrary to the results with Adh, we found that modules in merge/split families tend to have lower substitution rates than average. We believe that this discrepancy may be attributed to limitations in our model. For example, we computed rates across entire module trees, but it may be more appropriate to compute separate rates for portions of the tree affected and not affected by the merge/split event. We also did not consider differences in composition across species, which may confound the gene-specific and species-specific rate. Finally, we question whether an elevated substitution rate is indeed expected, since it is also plausible that modules should be more conserved after merge/split events in order to maintain functionality; that is, modules that undergo many substitutions may lose functionality and degenerate into pseudogenes.

9 GO term and Pfam domain enrichment/depletion

Enrichment/depletion values were computed using GOseq (Young et al. 2010) to correct for possible length (Supplemental Figure S4) and substitution bias (Supplemental Section 8, Supplemental Figure S5) in the data. Briefly, GOseq determines a probability weighting function that quantifies how the probability of a gene selected out of the reference set changes as a function of some external variable such as transcript length. It then resamples the genes many times, weighting the probability of choosing a gene using this function, and uses the resulting sampling distribution to calculate a p-value. Alternatively, GOseq uses the Wallenius non-central hypergeometric distribution to approximate p-values; we use this approximation in our analysis. p-values were corrected separately for length bias and substitution bias; correction for substitution bias did not change the set of enriched/depleted terms, so only the correction for length bias is shown in the main text.

For gene functions, we looked at gene ontology (GO) annotations for D. melanogaster, as the other species have little to no GO annotation. Only GO terms with experimental evidence were retained, and a gene with a GO term was expanded to also include all parent GO terms. D. melanogaster genes contain 4524 unique GO terms, 3327 of which are found in genes that participate in architecture families with merges or splits. Enrichment/depletion values were computed separately for each of the three ontologies biological process, cellular component, and molecular function. For domains, we looked at Pfam domains for all species. The Drosophila clade contains 3204 unique Pfam domains, 1510 of which are found in genes that participate in architecture families with merges or splits. After correcting for length bias, no Pfam domains were significantly enriched or depleted (hypergeometric test, \( p < 0.001 \), FDR correction).

10 Functional complementarity using DroID database

Rather than looking at shared GO terms, we searched for gene partners against the Drosophila interactions database (DroID) (Yu et al. 2008) (April 2010 release). Of the 1222 gene partners, 589 are those in which both genes have at least one known PPI (but not necessary with each other) and 135 (22.9% of 589) are those in which the genes are known to interact with each other. This is compared to 0.3% random (fold = 75.74, \( p < 0.001 \)).

Using a set of high confidence interactions in which we retain only PPI with experimental evidence (discarding those detected through homology), these numbers reduced to 57 gene partners in which both genes have at least one known PPI, 24 (42.1%) in which the genes are known to interact with each other, and 0.38% random (fold = 110.53, \( p < 0.001 \)). However, low counts means that we must take care in making any biological statements.
11 PPI in conservative set of architecture families

Using the conservative set of architecture families, we identified 222 gene partners within D. melanogaster. Of these, 33 gene partners have both genes annotated with GO terms, and 30 (90.9% of 33) share at least one GO term, compared to 61.8% random (fold = 1.47, \(p < 0.001\)). Using the DroID database, 140 gene partners are those in which both genes have at least one known PPI and 48 (34.3%) are those in which the genes are known to interact with each other, compared to 61.8% random (fold = 107.05, \(p < 0.001\)). Using the set of high confidence interactions, these numbers reduced to 16 gene partners in which both genes have at least one known PPI, 5 (31.3%) in which the genes are known to interact with each other, and 0.37% random (fold = 83.48, \(p < 0.001\)).

12 Detection of frameshift mutations

To investigate how often nucleotide similarity is not propagated to peptide similarity, we ran pairwise all-vs-all tBLASTx between the species, then post-processed the alignments with LALIGN and filters (e-value \(\leq 1 \times 10^{-5}\) and percent identity \(\geq 80\%\); a higher threshold for percent identity was used to account for the smaller nucleotide alphabet). We found 2740 hits at the nucleotide level but not at the peptide level, 130,422 hits at both the nucleotide and peptide level, and 345,554 hits at the peptide level but not at the nucleotide level. This translated to 1429 additional genes that may participate in a merge/split architecture family; note that the actual number of additional genes that participate in merge/split families is likely smaller than this count. For example, some of the genes may be part of families in which each gene consists of a single module, and the module is simply frameshifted in some genes, or some of the frameshift mutations may be a result of frameshift sequencing errors.

13 Systematic detection of gene fusion and fission by mechanism

13.1 Fusion/fission of adjacent genes

Two modules that merge or split were tagged if they were found in neighboring genes and the modules would be adjacent if the neighboring genes were considered as a single gene. This list was expanded to genes by looking for all genes with these tagged modules and including only those genes that are neighboring or are found in species descended from the most ancestral branch with a merge or split of the modules. Note that this list of genes includes both parental and children genes (e.g. pre- and post-merge/split genes), as this allows for ambiguities that may have arisen from ties in the MP reconstruction (e.g. one MP reconstruction finds split genes at the root followed by a merge along one branch, whereas another reconstruction finds a merged gene at the root followed by a split along the other branch). We also tested for experimental support for each gene by looking at EST and mRNA-seq data. Using our previously determined set of spanning ESTs (mRNA-seqs) (Supplemental Section 7.2), we called a gene consistent if all of the associated fragmented genes had EST (mRNA-seq) evidence but none were part of a spanning EST (mRNA-seq), inconsistent if at least one of the associated fragmented genes was part of a spanning EST (mRNA-seq), and unknown otherwise.

Genes that may have resulted from large-loop mismatch repair and replication slippage were detected by looking for merged genes flanked by (but not necessarily next to) two genes, with one gene containing one of the merged modules and the other gene containing the other merged module.

13.2 Retrotransposition and exon shuffling

While retrotransposition and exon shuffling are two separate mechanisms for novel gene formation, it has been suggested that retrotransposition is a driving mechanism for mediating exon shuffling (Gilbert et al. 1995), and exon shuffling by retrotransposition is one method for conferring novel gene functions to the resulting chimeric gene rather than allowing the retrosequence to degenerate into a pseudogene (Long 2001). To find retrotransposed-mediated exon shuffling events, we searched for modules that underwent a merge or split and that have multiple exons in at least one gene with merged form and a single exon in at least one gene with split form. This ignores possible retrotranspositions of single-exon genes which cannot be distinguished.
in our analysis from simple duplication. Exon comparison was performed at the module level to allow for
(1) a chimeric gene to result from a multi-exon parent fusing with a second retrotransposed parent, and (2)
partial gene retrotransposition. Furthermore, each merge or split formed a bifurcation of the species tree,
and the merged form and split form must belong to different sub trees formed from this bifurcation. Only
genes where the module has a single exon were included in the final count.

13.3 Duplication-degeneration

We constructed syntenic modules by defining a syntenic block to be at least three genes within 200 kb of
each other with no other blocks in between. Two genes are the result of duplication-degeneration if (1) they
result from a split, and (2) they belong to different syntenic blocks but have hits to the same syntenic block.
False detection could occur due to faulty syntenic blocks. For example, a missed connection between two
scaffolds may result in one syntenic block being separated into two blocks. As a simple test, we checked
whether both genes are located at the ends of their respective scaffolds; 36 modules and 50 genes remain.
Note that this method of detection does not take into account the case of undercollapsed assemblies where
genes are located within the scaffolds.
Supplementary Figures and Tables

**STAR-MP ALGORITHM**

1. **Traversals from leaves to root**
   - For each ancestral node, determine the possible architectures and their minimum costs-to-go.

2. **Termination**
   - Select the minimum cost architecture at the root.

3. **Traceback from root to leaves**
   - Assign architectures at the ancestral nodes and events along the branches.

1.1. **Generate possible architectures.**
   - Available modules: D+M, D+2S
   - Possible architectures: D+M, D+2S, (pruned)

1.2. **Determine (immediate) cost of architecture rearrangements.**
   - parent: child
   - D+M: 0, S+D: S

1.3. **Determine minimum cost-to-go of assigning parent architecture at node.**

**Figure S1: STAR-MP algorithm.** The forward phase of the STAR-MP algorithm performs a post-order traversal of the tree (step 1), considering at each stage a triplet of one parent node and two child nodes. The available parent modules (provided as input and found in our pipeline through reconciled module trees) and the possible children architectures (provided as input if the child is a leaf node or found recursively if the child is an ancestral node) are known. The possible parent architectures are generated through set permutations and pruned heuristically, and the minimum costs-to-go of assigning a parent architecture is determined (see inset at bottom, where the parent node under question is denoted by the ‘?’). The children architectures and events along the branches that led to this minimum cost-to-go is also retained for later traceback. This is repeated until the root of the tree is reached (step 2), at which point the minimum cost architecture is assigned to the root. The backward phase then performs a pre-order traversal of the tree (step 3) to assign the ancestral architectures and events.
Figure S2: Sensitivity and precision of STAR-MP in simulation under various event rates. See Figure 4 for details. Sensitivity decays dramatically as event rates increase, as is consistent with a conservative MP reconstruction. In contrast, precision is robust to event rates. Event rate multipliers were obtained from the simulations and differ from the input rate multipliers of N: (1,1,1,1), MS2: (1,1,2,2), MS4: (1,1,4,4), all2: (2,2,2,2), and all4: (4,4,4,4) due to our approach of discarding events that were impossible with the given starting architecture.

Figure S3: Correlation of module and exon boundaries. See Figure 5 for details.

Figure S4: Length bias. Genes in merge/split families tend to be longer than average. (Left) The distribution of gene lengths, bin size = 50 aa. Median lengths for all genes and for genes in merge/split families were 361 aa and 492 aa, respectively; the distributions differed significantly (Mann-Whitney U = 3.22 × 10^9, p < 2.23 × 10^{-308}, one-sided). (Right) The distribution of median gene lengths per architecture family, bin size = 50 aa. Median lengths for all families and for merge/split families were 391 aa and 524 aa, respectively; the distributions differed significantly (Mann-Whitney U = 3.19 × 10^7, p = 9.23 × 10^{-116}, one-sided).
Figure S5: No substitution rate bias. Genes in merge/split families have similar substitution rates compared to average. (Left), The distribution of substitution rates across module families, bin size = 0.1 sub/site. Median rates for all modules and for modules in merge/split families were 0.723 sub/site and 0.701 sub/site, respectively; the distributions differed significantly (Mann-Whitney U = 1.35 × 10^8, p = 5.23 × 10^{-10}, one-sided). (Right), The distribution of median substitution rates per architecture family, bin size = 50 aa. Median rates for all families and for merge/split families were 0.744 sub/site and 0.725 sub/site, respectively; the distributions differed significantly (Mann-Whitney U = 2.85 × 10^7, p = 1.73 × 10^{-4}, one-sided).

Figure S6: Distribution of inferred event counts across architecture families. Event counts are not dominated by a subset of families. Most families have at most one event per type along each branch.
Figure S7: Distribution of inferred evolutionary events using a conservative set of architecture families. See Figures 7 and S6 for details.
Figure S8: The inferred evolutionary history of GA28694 in *D. pseudoobscura* from CG4617 and frizzled 4 (*fz4*). (A) The MP architecture scenario. Architectures 00821 and 05692-03769 are usually in separate genes corresponding to CG4617 and *fz4* orthologs. However, *dpse* contains the fused architecture 00821-03769 in gene GA28694. (B) Along the *dpse* branch, the MP reconstruction infers the loss of module 22266 and duplications of modules 00821 and 03769 followed by a merge to form 00821-03769. A (hidden) split is also inferred between 05692 and 03769 due to the sub-architecture duplication of 03769 from *fz4* (05692-03769). (C) A genome level view shows that the orthologs of CG4617 and *fz4* are adjacent in all species except *dpse*. In *dpse*, the new, fused gene GA28694 is located between CG4617 and *fz4*. (D) The inferred evolutionary history of GA28694 in *dpse* through duplication and exon shuffling. We postulate that GA28694 was formed by tandem duplication of the chromosomal region from CG4617 to *fz4*, followed by either large-loop mismatch repair or replication slippage to form the merged gene GA28694. Duplication allows *dpse* to retain the original gene functions of CG4617 and *fz4* and gain a new function with GA28694. Note that while module 05692 is duplicated as part of the tandem duplication, it is subsequently lost in *dpse*; thus, the MP reconstruction does not infer a duplication of 05692.
Figure S9: The inferred evolutionary history ofjingwei (jgw) in D. yakuba through retrotransposition and exon shuffling of alcohol dehydrogenase (ADH) and yellow emperor (ymp). (A) The MP architecture scenario. (1) Along the branch leading the dyak, the MP reconstruction infers the duplication of modules 17143 and 00370 followed by their subsequent merge to form jgw. (2) Along branch leading the dmel, we see an interaction between modules 17143 and 17258 to form ymp. However, there is strong evidence that the gene pairs GE10684-GE10685 and GG12235-GG12236 are actually single gene orthologs of ymp. These gene break errors lead to an incorrectly inferred ancestral architecture for ymp in which the modules 17143 and 17258 are found in separate genes rather than fused in a single gene. (3) ADH consists of a single module 00370, and there are multiple copies of this module in isolated form in many genomes. Multiple cases of fusions with ADH-derived modules have also been found experimentally, suggesting that ADH may be enriched for fusion events. We find module 00370 fused to the architecture 03541-01876 in dpse, which has not been previously found in literature. It is possible that GA25237 and GA25238 are further examples of ADH-derived chimeric genes. (B) A genome level view of ymp reveals gene break errors in the ymp orthologs. This is supported by exon structure and genome alignment and partially supported by EST evidence: ymp in dmel has multiple full ESTs (e.g. ESTs span entire gene), and GE10684 and GE10685 in dyak have multiple spanning ESTs, but none of GG12235 or GG12236 in dere nor GF17267 in dana have ESTs. ADH is found on a different chromosome (scaffold). (C) The inferred evolutionary history of jgw in dyak. ymp is first duplicated to create a second copy yande (Long et al. 2003), then ADH is retrotransposed between the third and fourth exons of yande followed by degeneration of the yande exons found after the insertion point. Exons in gray represent exons that are not part of the longest transcript.
<table>
<thead>
<tr>
<th>species</th>
<th># genes</th>
<th># genes w/ EST (mRNA-seq)</th>
<th># gene pairs w/ EST (mRNA-seq)</th>
<th># gene pairs w/ spanning EST (mRNA-seq)</th>
<th>error rate (%) of EST (mRNA-seq)</th>
<th># families w/ EST (mRNA-seq)</th>
<th># families w/ spanning EST (mRNA-seq)</th>
<th>error rate (%) of EST (mRNA-seq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmel</td>
<td>4769</td>
<td>4439 (4455)</td>
<td>420</td>
<td>348 (333)</td>
<td>18 (10)</td>
<td>5.17 (3.00)</td>
<td>284 (296)</td>
<td>18 (10)</td>
</tr>
<tr>
<td>dyak</td>
<td>5604</td>
<td>429</td>
<td>806</td>
<td>18</td>
<td>4</td>
<td>22.22</td>
<td>17</td>
<td>4</td>
</tr>
<tr>
<td>dere</td>
<td>5169</td>
<td>1697</td>
<td>715</td>
<td>7</td>
<td>11.86</td>
<td>55</td>
<td>7</td>
<td>12.73</td>
</tr>
<tr>
<td>dana</td>
<td>4988</td>
<td>1804</td>
<td>655</td>
<td>11</td>
<td>11.96</td>
<td>77</td>
<td>11</td>
<td>14.29</td>
</tr>
<tr>
<td>dpse</td>
<td>5092</td>
<td>935 (4587)</td>
<td>618</td>
<td>30 (496)</td>
<td>6 (16)</td>
<td>20.00 (3.23)</td>
<td>23 (413)</td>
<td>6 (16)</td>
</tr>
<tr>
<td>dwil</td>
<td>4827</td>
<td>1592</td>
<td>507</td>
<td>7</td>
<td>10.14</td>
<td>58</td>
<td>7</td>
<td>12.07</td>
</tr>
<tr>
<td>dmoj</td>
<td>4797</td>
<td>1713 (4507)</td>
<td>640</td>
<td>62 (575)</td>
<td>10 (36)</td>
<td>16.13 (6.26)</td>
<td>55 (465)</td>
<td>10 (36)</td>
</tr>
<tr>
<td>dvir</td>
<td>4809</td>
<td>1847</td>
<td>654</td>
<td>81</td>
<td>12</td>
<td>14.81</td>
<td>77</td>
<td>12</td>
</tr>
<tr>
<td>dgri</td>
<td>5227</td>
<td>1777</td>
<td>651</td>
<td>75</td>
<td>3</td>
<td>4.00</td>
<td>64</td>
<td>3</td>
</tr>
<tr>
<td>total</td>
<td>45,282</td>
<td>16,233 (13,549)</td>
<td>5666</td>
<td>834 (1404)</td>
<td>78 (62)</td>
<td>9.35 (4.42)</td>
<td>451 (766)</td>
<td>52 (51)</td>
</tr>
</tbody>
</table>

Table S1: EST evidence for genes in merge/split families. See Table 1 for details. Similar statistics are also provided across architecture families, where at least one gene pair must be present in the architecture family for it to be included in the count. Large error rates can be attributed to low counts.
Table S2: EST and mRNA-seq evidence for fusion and fission events. The number of consistent, inconsistent, and unknown scenarios and events based on EST and mRNA-seq evidence. This data excludes the 249 families without merges or splits in the reconstruction.

<table>
<thead>
<tr>
<th>species</th>
<th>EST</th>
<th>mRNA-seq</th>
<th>EST</th>
<th>mRNA-seq</th>
<th>EST</th>
<th>mRNA-seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmel</td>
<td>3858</td>
<td>583</td>
<td>619</td>
<td>16.0%</td>
<td>48</td>
<td>1.2%</td>
</tr>
<tr>
<td>dyak</td>
<td>4876</td>
<td>1130</td>
<td>1992</td>
<td>40.9%</td>
<td>54</td>
<td>1.1%</td>
</tr>
<tr>
<td>dere</td>
<td>5659</td>
<td>1400</td>
<td>1776</td>
<td>31.4%</td>
<td>32</td>
<td>0.6%</td>
</tr>
</tbody>
</table>

Table S3: Correlation of module boundaries and domain boundaries with exon boundaries. For each species, the total number of proteins, exons, modules, and domains is provided. Furthermore, the observed and expected number of exon-bordering module (the number of modules in which both boundaries are within ±10 aa of an exon) and the fold percentage are provided. To calculate the expected number of exon-bordering modules, we derived the probability $P$ of an exon border falling onto any amino acid by dividing the total number of exon borders by the total length of proteins. We also determined the total number $T$ of amino acids within ±10 aa of any module boundary. Based on a null hypothesis of randomly distributed exon borders, the product $PT$ gives the expected number of exon borders within ±10 aa of a module boundary; thus, the expected number of exon-bordering modules is $E = (P T / \# \text{exons})^2 / \# \text{modules}$. The same analysis is performed exchanging module with domains. $P$-values were calculated based on a chi-square distribution (dof = 1), and all $p$-values satisfied $p < 2.23 \times 10^{-308}$. Column sums may not equal total due to rounding of the expected value.

<table>
<thead>
<tr>
<th>species</th>
<th>0-0 O-E</th>
<th>0-1 O/E</th>
<th>0-2 O/E</th>
<th>sym O-E</th>
<th>non-sym O/E</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmel</td>
<td>9560</td>
<td>2071</td>
<td>4.62</td>
<td>67</td>
<td>1226</td>
</tr>
<tr>
<td>dyak</td>
<td>10779</td>
<td>2315</td>
<td>4.66</td>
<td>48</td>
<td>1340</td>
</tr>
<tr>
<td>dere</td>
<td>10168</td>
<td>2187</td>
<td>4.65</td>
<td>57</td>
<td>1269</td>
</tr>
<tr>
<td>dana</td>
<td>8594</td>
<td>1880</td>
<td>4.57</td>
<td>44</td>
<td>1077</td>
</tr>
<tr>
<td>dpse</td>
<td>8434</td>
<td>1807</td>
<td>4.67</td>
<td>49</td>
<td>1075</td>
</tr>
<tr>
<td>dwil</td>
<td>7876</td>
<td>1708</td>
<td>4.61</td>
<td>46</td>
<td>1026</td>
</tr>
<tr>
<td>dmoj</td>
<td>8978</td>
<td>1929</td>
<td>4.65</td>
<td>52</td>
<td>1104</td>
</tr>
<tr>
<td>dvir</td>
<td>9110</td>
<td>1942</td>
<td>4.69</td>
<td>61</td>
<td>1148</td>
</tr>
<tr>
<td>dgri</td>
<td>9225</td>
<td>1990</td>
<td>4.64</td>
<td>52</td>
<td>1168</td>
</tr>
</tbody>
</table>

Table S4: Introns phases of exon-bordering modules. For each species, we list the observed and expected numbers of modules with the given intron-phase combinations, where a module is labeled with the phases of its flanking introns. The expected numbers were calculated as in (Long et al. 2003). Specifically, assuming that any two introns can flank an module, the expected number of modules with intron-phase $(i, j)$ is given by $E_{ij} = P_i P_j N$, where $0 \leq i, j \leq 2$, $P_i$ is the proportion of intron phase $i$ actually observed, and $N$ is the total observed number of intron associations. $P$-values were also calculated based on a chi-square distribution (dof = 1), and all $p$-values satisfied $p < 2.23 \times 10^{-308}$. Column sums may not equal total due to rounding of the expected value.
### Table S5: Intron phases of exon-bordering domains.

See Table S4 for details. All p-values satisfied $p < 1 \times 10^{-5}$ except for the 1-1 domains.

<table>
<thead>
<tr>
<th>species</th>
<th>0-0</th>
<th>1-1</th>
<th>2-2</th>
<th>sym</th>
<th>non-sym</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O/E</td>
<td>O/E</td>
<td>O/E</td>
<td>O/E</td>
<td>O/E</td>
</tr>
<tr>
<td>dmel</td>
<td>668</td>
<td>246</td>
<td>2.71</td>
<td>149</td>
<td>146</td>
</tr>
<tr>
<td>dyak</td>
<td>745</td>
<td>262</td>
<td>2.85</td>
<td>143</td>
<td>152</td>
</tr>
<tr>
<td>dere</td>
<td>693</td>
<td>250</td>
<td>2.78</td>
<td>137</td>
<td>145</td>
</tr>
<tr>
<td>dana</td>
<td>770</td>
<td>275</td>
<td>2.80</td>
<td>139</td>
<td>157</td>
</tr>
<tr>
<td>dpse</td>
<td>784</td>
<td>279</td>
<td>2.81</td>
<td>164</td>
<td>156</td>
</tr>
<tr>
<td>dwil</td>
<td>750</td>
<td>269</td>
<td>2.79</td>
<td>149</td>
<td>162</td>
</tr>
<tr>
<td>dmoj</td>
<td>703</td>
<td>250</td>
<td>2.82</td>
<td>147</td>
<td>152</td>
</tr>
<tr>
<td>dvir</td>
<td>734</td>
<td>258</td>
<td>2.85</td>
<td>139</td>
<td>152</td>
</tr>
<tr>
<td>dgri</td>
<td>809</td>
<td>284</td>
<td>2.84</td>
<td>165</td>
<td>167</td>
</tr>
<tr>
<td>total</td>
<td>6656</td>
<td>2372</td>
<td>2.81</td>
<td>1332</td>
<td>1389</td>
</tr>
</tbody>
</table>

### Table S6: Lineage-specific merge and split events.

<table>
<thead>
<tr>
<th>species</th>
<th>dist</th>
<th>M</th>
<th>S</th>
<th>genes$^a$</th>
<th>% dist$^b$</th>
<th>% M (ratio)$^c$</th>
<th>% S (ratio)$^c$</th>
<th>% dist,$^d$</th>
<th>% M (ratio)$^e$</th>
<th>% S (ratio)$^e$</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmel</td>
<td>11.23</td>
<td>457</td>
<td>233</td>
<td>446</td>
<td>2.9</td>
<td>9.4 (3.2)</td>
<td>4.1 (1.4)</td>
<td>3.7</td>
<td>16.3 (4.4)</td>
<td>5.2 (1.4)</td>
</tr>
<tr>
<td>dyak</td>
<td>8.57</td>
<td>268</td>
<td>634</td>
<td>341</td>
<td>2.2</td>
<td>5.5 (2.5)</td>
<td>11.2 (5.1)</td>
<td>2.8</td>
<td>9.6 (3.4)</td>
<td>14.2 (5.0)</td>
</tr>
<tr>
<td>dere</td>
<td>8.57</td>
<td>233</td>
<td>340</td>
<td>341</td>
<td>2.2</td>
<td>4.8 (2.2)</td>
<td>6.0 (2.7)</td>
<td>2.8</td>
<td>8.3 (2.9)</td>
<td>7.6 (2.7)</td>
</tr>
<tr>
<td>dana</td>
<td>53.40</td>
<td>317</td>
<td>510</td>
<td>335</td>
<td>13.8</td>
<td>6.5 (0.5)</td>
<td>9.0 (0.7)</td>
<td>7.6</td>
<td>11.3 (0.6)</td>
<td>11.4 (0.6)</td>
</tr>
<tr>
<td>dpse</td>
<td>55.80</td>
<td>362</td>
<td>523</td>
<td>319</td>
<td>14.4</td>
<td>7.4 (0.5)</td>
<td>9.2 (0.6)</td>
<td>18.4</td>
<td>12.9 (0.7)</td>
<td>11.7 (0.6)</td>
</tr>
<tr>
<td>dwil</td>
<td>62.49</td>
<td>309</td>
<td>540</td>
<td>318</td>
<td>16.1</td>
<td>6.3 (0.4)</td>
<td>9.5 (0.6)</td>
<td>20.6</td>
<td>11.0 (0.5)</td>
<td>12.1 (0.6)</td>
</tr>
<tr>
<td>dmoj</td>
<td>32.74</td>
<td>283</td>
<td>546</td>
<td>295</td>
<td>8.4</td>
<td>5.8 (0.7)</td>
<td>9.7 (1.1)</td>
<td>10.8</td>
<td>10.1 (0.9)</td>
<td>12.2 (1.1)</td>
</tr>
<tr>
<td>dvir</td>
<td>32.74</td>
<td>301</td>
<td>403</td>
<td>345</td>
<td>8.4</td>
<td>6.2 (0.7)</td>
<td>7.1 (0.8)</td>
<td>10.8</td>
<td>10.8 (1.0)</td>
<td>9.0 (0.8)</td>
</tr>
<tr>
<td>dgri</td>
<td>37.11</td>
<td>269</td>
<td>735</td>
<td>314</td>
<td>9.6</td>
<td>5.5 (0.6)</td>
<td>13.0 (1.4)</td>
<td>12.3</td>
<td>9.6 (0.8)</td>
<td>16.5 (1.3)</td>
</tr>
<tr>
<td>total</td>
<td>302.65</td>
<td>2799</td>
<td>4464</td>
<td>3044</td>
<td>77.9</td>
<td>57.4 (0.1)</td>
<td>78.9 (1.0)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

$^a$Number of fused genes for which the split form consists of two adjacent genes.

$^b$Branch length divided by the total branch length.

$^c$Number of merges or splits in this genome divided by the total number of merges or splits, and ratio of ($\% M$)/($\%$ dist) or ($\% S$)/($\%$ dist).

$^d$Branch length divided by the total leaf branch length.

$^e$Number of merges or splits in this genome divided by the total number of lineage-specific merges or splits, and ratio of ($\% M_l$)/($\%$ dist$_l$) or ($\% S_l$)/($\%$ dist$_l$).

### Table S7: GO enrichment for genes undergoing module rearrangement in conservative set of architecture families.

See Table 2 for details. Here, $n = 2506$ and $N = 12,408$, and we used a $p$-value cutoff of $p < .01$.

<table>
<thead>
<tr>
<th>rank</th>
<th>GO ID</th>
<th>GO term</th>
<th>k</th>
<th>m</th>
<th>fold</th>
<th>p-value</th>
<th>p-value</th>
<th>q-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GO:0007275</td>
<td>multicellular organismal develop</td>
<td>323</td>
<td>1119</td>
<td>1.43</td>
<td>$2.24 \times 10^{-14}$</td>
<td>$2.44 \times 10^{-6}$</td>
<td>$4.39 \times 10^{-3}$</td>
</tr>
<tr>
<td>2</td>
<td>GO:0032502</td>
<td>developmental process</td>
<td>355</td>
<td>1253</td>
<td>1.40</td>
<td>$2.02 \times 10^{-13}$</td>
<td>$3.41 \times 10^{-6}$</td>
<td>$4.39 \times 10^{-3}$</td>
</tr>
<tr>
<td>3</td>
<td>GO:0032501</td>
<td>multicellular organismal process</td>
<td>382</td>
<td>1358</td>
<td>1.39</td>
<td>$6.30 \times 10^{-14}$</td>
<td>$2.79 \times 10^{-6}$</td>
<td>$4.39 \times 10^{-3}$</td>
</tr>
<tr>
<td>MERGES</td>
<td>all</td>
<td>w/o generation</td>
<td>w/ generation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------</td>
<td>----------------</td>
<td>--------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of events</td>
<td>2559</td>
<td>1676 (65.5%)</td>
<td>883 (34.5%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>retained at least one split architecture</td>
<td>2392 (93.5%)</td>
<td>1620 (96.7%)</td>
<td>772 (87.4%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>retained both split architectures</td>
<td>1413 (55.2%)</td>
<td>1413 (84.3%)</td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SPLITS</th>
<th>all</th>
<th>w/o loss</th>
<th>w/ loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of events</td>
<td>2446</td>
<td>670 (27.4%)</td>
<td>1776 (72.6%)</td>
</tr>
<tr>
<td>retained merged architecture</td>
<td>465 (19.0%)</td>
<td>443 (66.1%)</td>
<td>22 (1.2%)</td>
</tr>
</tbody>
</table>

Table S8: Retainment of ancestral architectures by merge and split events in conservative set of architecture families. See Table 3 for details.

<table>
<thead>
<tr>
<th>species</th>
<th>dist</th>
<th>M</th>
<th>S</th>
<th>% dist</th>
<th>% M (ratio)</th>
<th>% S (ratio)</th>
<th>% dist</th>
<th>% M_L (ratio)</th>
<th>% S_L (ratio)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmel</td>
<td>11.23</td>
<td>174</td>
<td>122</td>
<td>2.9</td>
<td>6.8 (2.3)</td>
<td>4.2 (1.4)</td>
<td>3.7</td>
<td>13.7 (3.7)</td>
<td>5.2 (1.4)</td>
</tr>
<tr>
<td>dyak</td>
<td>8.57</td>
<td>110</td>
<td>309</td>
<td>2.2</td>
<td>4.3 (1.9)</td>
<td>10.7 (4.9)</td>
<td>2.8</td>
<td>8.7 (3.1)</td>
<td>13.5 (4.8)</td>
</tr>
<tr>
<td>dere</td>
<td>8.57</td>
<td>106</td>
<td>169</td>
<td>2.2</td>
<td>4.1 (1.9)</td>
<td>5.9 (2.7)</td>
<td>2.8</td>
<td>8.4 (3.0)</td>
<td>7.4 (2.6)</td>
</tr>
<tr>
<td>dana</td>
<td>53.40</td>
<td>146</td>
<td>271</td>
<td>13.8</td>
<td>5.7 (0.4)</td>
<td>9.4 (0.7)</td>
<td>17.6</td>
<td>11.5 (0.7)</td>
<td>11.9 (0.7)</td>
</tr>
<tr>
<td>dpse</td>
<td>55.80</td>
<td>170</td>
<td>248</td>
<td>14.4</td>
<td>6.6 (0.5)</td>
<td>8.6 (0.6)</td>
<td>18.4</td>
<td>13.4 (0.7)</td>
<td>10.9 (0.6)</td>
</tr>
<tr>
<td>dwil</td>
<td>62.49</td>
<td>146</td>
<td>293</td>
<td>16.1</td>
<td>5.7 (0.4)</td>
<td>10.2 (0.6)</td>
<td>20.6</td>
<td>11.5 (0.6)</td>
<td>12.8 (0.6)</td>
</tr>
<tr>
<td>dmoj</td>
<td>32.74</td>
<td>152</td>
<td>282</td>
<td>8.4</td>
<td>5.9 (0.7)</td>
<td>9.8 (1.2)</td>
<td>10.8</td>
<td>12.0 (1.1)</td>
<td>12.3 (1.1)</td>
</tr>
<tr>
<td>dvir</td>
<td>32.74</td>
<td>137</td>
<td>204</td>
<td>8.4</td>
<td>5.3 (0.6)</td>
<td>7.1 (0.8)</td>
<td>10.8</td>
<td>10.8 (1.0)</td>
<td>8.9 (0.8)</td>
</tr>
<tr>
<td>dgri</td>
<td>37.11</td>
<td>127</td>
<td>386</td>
<td>9.6</td>
<td>4.9 (0.5)</td>
<td>13.4 (1.4)</td>
<td>12.3</td>
<td>10.0 (0.8)</td>
<td>16.9 (1.4)</td>
</tr>
<tr>
<td>total</td>
<td>302.65</td>
<td>2067</td>
<td>2880</td>
<td>77.9</td>
<td>39.4 (0.6)</td>
<td>79.3 (1.0)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table S9: Lineage-specific merge and split events in conservative set of architecture families. See Table S6 for details.
<table>
<thead>
<tr>
<th>chimeric gene</th>
<th>parental genes</th>
<th>source citation</th>
<th>detected</th>
<th>reason not detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG10102</td>
<td>CG12505 (Arc1)</td>
<td>E</td>
<td>Z</td>
<td>only one hit satisfies %id thr</td>
</tr>
<tr>
<td>CG12592</td>
<td>CG12819 (slc), CG18545</td>
<td>E</td>
<td>R,Z</td>
<td>under-clustered module instances</td>
</tr>
<tr>
<td>CG18853</td>
<td>CG12822, CG11205 (phr)</td>
<td>E</td>
<td>R,Z</td>
<td>X</td>
</tr>
<tr>
<td>CG18217</td>
<td>CG17286 (spd-2), CG4098</td>
<td>E</td>
<td>R,Z</td>
<td>X</td>
</tr>
<tr>
<td>CG31352 (unc-115)</td>
<td>CG31352</td>
<td>E</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG31687</td>
<td>CG31688, CG2508 (cdc23)</td>
<td>E</td>
<td>R,Z</td>
<td>X</td>
</tr>
<tr>
<td>CG33105 (p24-2)</td>
<td>CG33104 (eca), CG31352</td>
<td>E</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG32745</td>
<td>CG3458 (Top3β)</td>
<td>E</td>
<td>Z</td>
<td>CG32745 became pseudogene</td>
</tr>
<tr>
<td>CG31864 (Qrtl)</td>
<td>CG5202 (escl), CG12264</td>
<td>E,R,Z</td>
<td></td>
<td>frameshift</td>
</tr>
<tr>
<td>CG32474 (Ubi-p5E)</td>
<td>CG11700</td>
<td>E</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG18789</td>
<td>CG18787</td>
<td>E</td>
<td>Z</td>
<td>only one hit</td>
</tr>
<tr>
<td>CG17706</td>
<td>CG4211 (nonA)</td>
<td>E</td>
<td>Z</td>
<td>only one hit; CG17706 became CG40282</td>
</tr>
<tr>
<td>CG32788 (Crg-1)</td>
<td>CG12632 (fd3F)</td>
<td>N</td>
<td>Z</td>
<td>only one hit</td>
</tr>
<tr>
<td>CG41454</td>
<td>CG33217</td>
<td>N</td>
<td>Z</td>
<td>only one hit</td>
</tr>
<tr>
<td>CG41054</td>
<td>CG4849</td>
<td>N</td>
<td>Z</td>
<td>only one hit</td>
</tr>
<tr>
<td>CG23218</td>
<td>CG9191 (Klp61F), CG9187 (psf1)</td>
<td>N,R,Z</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CG32821</td>
<td>CG12655</td>
<td>R</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG40100</td>
<td>CG30022</td>
<td>U</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG33221</td>
<td>CG33213</td>
<td>R</td>
<td>Z</td>
<td>only one hit satisfies %id thr</td>
</tr>
<tr>
<td>CG32733</td>
<td>CG9821</td>
<td>E</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG12184</td>
<td>CG12179</td>
<td>U</td>
<td>Z</td>
<td>only one hit satisfies %id thr</td>
</tr>
<tr>
<td>CG12824</td>
<td>CG12925</td>
<td>R</td>
<td>Z</td>
<td>no hits</td>
</tr>
<tr>
<td>CG14810</td>
<td>CG14911</td>
<td>R</td>
<td>Z</td>
<td>only one hit</td>
</tr>
<tr>
<td>CG32584</td>
<td>CG15645 (cerv)</td>
<td>N</td>
<td>Z</td>
<td>CG32583 split into CG42299/CG42300</td>
</tr>
<tr>
<td>CG17797 (Acp29AB)</td>
<td>CG15818</td>
<td>E</td>
<td>Z</td>
<td>no hits satisfy %id thr</td>
</tr>
<tr>
<td>CG17440</td>
<td>CG17446</td>
<td>R</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG13794</td>
<td>CG33296</td>
<td>N</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG6900</td>
<td>CG6891</td>
<td>R</td>
<td>Z</td>
<td>only one hit</td>
</tr>
<tr>
<td>CG14666 (Tim17a2)</td>
<td>CG10090 (Tim17a1)</td>
<td>U</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG32071</td>
<td>CG12522 (Gtp-bp)</td>
<td>R</td>
<td>Z</td>
<td>no hits</td>
</tr>
<tr>
<td>CG14628</td>
<td>CG18823</td>
<td>U</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG17472</td>
<td>CG31680</td>
<td>U</td>
<td>Z</td>
<td>only one hit</td>
</tr>
<tr>
<td>CG33317</td>
<td>CG3584 (qkr58E-3)</td>
<td>R</td>
<td>Z</td>
<td>CG33317 became pseudogene</td>
</tr>
<tr>
<td>CG3410 (lectin-24A)</td>
<td>CG7106 (lectin-28C)</td>
<td>U</td>
<td>Z</td>
<td>no hits satisfy %id thr</td>
</tr>
<tr>
<td>CG9902</td>
<td>CG7692</td>
<td>U</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG10991</td>
<td>CG9360</td>
<td>R</td>
<td>Z</td>
<td>no hits satisfy %id thr</td>
</tr>
<tr>
<td>CG7804</td>
<td>CG10327 (TBPH)</td>
<td>U</td>
<td>Z</td>
<td>only one hit satisfies %id thr</td>
</tr>
<tr>
<td>CG9906</td>
<td>CG19598 (Cnx99A)</td>
<td>R</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG11235</td>
<td>CG19598 (Cnx99A)</td>
<td>N</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>CG31904</td>
<td>CG13796, CG7216 (Arp1)</td>
<td>E</td>
<td>R</td>
<td>X</td>
</tr>
<tr>
<td>CG30457</td>
<td>CG10953, CG31705</td>
<td>E</td>
<td>R</td>
<td>no hits</td>
</tr>
<tr>
<td>CG17196</td>
<td>CG17197, CG31705</td>
<td>E</td>
<td>R</td>
<td>no hits satisfy %id thr</td>
</tr>
<tr>
<td>CG11961</td>
<td>CG9416, CG30049</td>
<td>E</td>
<td>R</td>
<td>X</td>
</tr>
<tr>
<td>CG3978 (pnr)</td>
<td>CG9656 (grm), CG10278 (GATAe)</td>
<td>E</td>
<td>R</td>
<td>X</td>
</tr>
<tr>
<td>CG6844 (nAcRα-96Ab)</td>
<td>CG5610 (nAcRα-96Aa), CG11348 (nAcRβ-64B)</td>
<td>E</td>
<td>R</td>
<td>X</td>
</tr>
<tr>
<td>CG6653 (Ugt86De)</td>
<td>CG31082 (Gga), CG17200 (Ugt86Dg)</td>
<td>E</td>
<td>R</td>
<td>overlapping alignments</td>
</tr>
<tr>
<td>CG31668</td>
<td>CG33124, CG8451</td>
<td>E</td>
<td>R</td>
<td>overlapping alignments</td>
</tr>
</tbody>
</table>

**Table S10**: Detection of previously identified chimeric genes. Chimeric sources are divided into ‘E’ (from exons of another parental gene), ‘N’ (from intron or intergenic module), ‘R’ (simple tandem repeats or repetitive elements) and ‘U’ (unknown sources). Citations are ‘R’ (Rogers et al. 2009), and ‘Z’ (Zhou et al. 2008). Hits refer to LALIGN hits for the chimeric gene, e.g. only one LALIGN hit for the chimeric gene satisfied the percent identity threshold.
Online Supplementary Files

1 Analysis of 9 Drosophila genomes


flies9.ids.txt: Gene names.

regs.tar.gz: Modules and module families.

fams.txt, fams.ms.txt, fams.ms.cons.txt: Architecture families, “merge/split” architecture families, and conservative “merge/split” architecture families.

fams.ms.tar.gz: For each architecture family, the extant architectures, bootstrapped gene trees, reconstructed architecture scenario, and a figure of the reconstructed architecture scenario.

2 Catalog of genes grouped by mechanism of formation

nbrs.txt, nbrs_genes.txt: Genes involved in fusion/fission of neighboring genes (either as a parent or child), and whether they are supported by experimental evidence.

dupmerge.txt: Genes involved in large-loop mismatch repair or replication slippage.

retro.txt: Genes involved in fusion/fission via retrotransposition.

dupdeg.txt: Genes involved in duplication-degeneration.
References


