APPENDIX 2 - SOLANACEAE PROJECT SEQUENCING AND BIOINFORMATICS STANDARDS AND GUIDELINES

NOTE: FEEDBACK IS VERY WELCOME. PLEASE CONTACT LUKAS MUELLER (LAM87@CORNELL.EDU) FOR CHANGES AND SUGGESTIONS.

Coordinators:
Lukas Mueller, SGN
Klaus Mayer, MIPS

Based on documents from the Medicao Sequencing Project kindly provided by Nevin Young, and a document obtained from Gramene, kindly provided by Pankaj Jaiswal, and the GMOD project (http://www.gmod.org).

This document is an attempt to develop a series of standards and guidelines for the SOL tomato sequencing project to ensure maximum consistency of tomato genome assembly and annotation across different centers. It addresses sequencing quality standards, BAC assembly, SOL data release policy, gene naming (traditional gene names, BAC based names, chromosome based names and gene classification), gene structural and functional annotation, data exchange formats and guidelines for software systems and platforms, with some of the sections still a bit patchy. It should represent a realistic and reasonable compromise that all project members can agree on, so the contributions of all involved with SOL are highly welcome. The standards and guidelines are based mostly on those developed for other genomes, where applicable, as noted in the text. This should make the results of this project more readily comparable to other projects using similar standards.

TABLE OF CONTENTS:

I. Standards for BAC Closure/Finishing

The standards for BAC closure/finishing have been adapted from the Medicago sequencing project. As new genomes are sequenced, it is clear that it is neither cost-effective in terms of gene discovery nor biological impact to finish every BAC to HTGS Phase 3. Guided by newer finishing techniques and evolving standards recently adopted by the NHGRI, NSF, and other funding agencies, we propose similar minimum finishing standards for the Tomato genome project. We will use the following suite of standard

techniques to close as many gaps in BAC clones remaining after sequencing and assembly, with a goal of reducing overall error rate to fewer than one uncertain base per 10,000 bases.

• Re-sequencing short reads if they extend a contig.
• Re-sequencing failed mates if they fall into gaps.
• Re-sequencing and editing individual reads to higher accuracy.
• Additional reads, if deemed necessary, to improve the sequence quality of low accuracy regions
• Re-assembly at higher stringency.
• Primer walking directly off shotgun sub-clones or PCR amplified region.
• Modified dye primers.
• PCR across putative gaps to order and orient assemblies and, if possible, obtain the sequence of gaps spanned by these PCR products.
• Other techniques, including the use of modified nucleotides, enzymes, and chemicals, to read through GC-rich regions/hard stops may be attempted, but not repeatedly.

Extraordinary, labor-intensive techniques, such as transposon-base sequencing and micro (shatter) libraries may be employed, but not extensively. Thus, while we will make every reasonable effort to close all BACs, we recognize that not all BACs will reach HTGS 3. In those instances, regions approaching HTGS 3 will be fully annotated as such.

We propose the following minimal standards for "finished" sequence — With exceptions noted in annotation comments:
1. A single contig is generated.
2. The bulk of the sequence should be derived from multiple subclones sequenced from both strands. Less than 3% of the sequence should be derived from multiple subclones sequenced from the same strand with the same chemistry. These regions must pass manual inspection for any sequence problems, but do not need to be annotated unless the sequence quality falls below phred 30. Less than 1% of the sequence should be derived from a single subclone. In the case of a region covered by a single subclone, the clone must be sequenced either on both strands or with two different chemistries, and the region must be annotated.
3. More than 99% of the sequence has less than one error in 10,000 bp as reported by phrap or other sequence assembly consensus scores. Exceptions must be manually checked and have passed inspection for possible problems. These areas must be annotated.
4. The assembled sequence is confirmed by restriction enzyme digestion.
5. Where gap closure/finishing is difficult to complete, sequences should be submitted to GenBank as HTGS phase 2 (contigs ordered and oriented) with exceptions noted.
6. Estimates of the size(s) of any un-sequencable gaps should be noted.

Repeats
• In cases of simple repeat sequences, including single nucleotide repeats where the number of repeats can not be determined, the length of the repeat region should be estimated by restriction enzyme digestion or PCR.
• If large repeats cannot be resolved, the size of the repeat region, confirmed by restriction enzyme digestion or PCR, the nature of the repeats, the size of repeats, and the finishing problem should be indicated.
• Sequences of bacterial transposons and other obvious contaminants should be screened and deleted from the finished sequence; the size, sequence, and position of the deleted region should be indicated.
• Sequence should also be screened against chloroplast and mitochondrial sequence. Arabidopsis sequences may be used for that purpose. Criteria for determining whether the sequences represent real insertions into the chromosomal sequence have to be established. [Suggestions welcome].

• For BAC overlaps, we will aim for overlaps of 5-10 kb and will plan to finish to high quality (phase 3) at least 5 kb in the BAC that forms the overlap (ie joins second). We will also accept smaller overlaps (down to ~100 bases). However, such overlaps should be supported by fingerprints from other overlapping BACs and by PCR across the junctions.
• We will develop protocols and XML/DTDto document evidence supporting BAC overlaps. This will include noting the region of high quality overlap (100% identity) plus any discrepancies.
• Since one BAC will be finished before the other, in cases where the BAC overlap is between BACs finished by two different groups, the "owner" of the second BAC will be responsible for generating the data describing the BAC overlap. These data will be maintained both in association with the individual BAC annotation and in the centralized project database at Cornell, where it will be immediately available to both sequencing centers and the community at large.

Finishing Chromosomes and Defining Gaps on Chromosomes

We will be sequencing chromosomes starting from contigs that contain marker-anchored seed BACs. Contigs will be extended by a combination of BAC-end and fingerprint data. Gaps will be apparent when no clones extending contigs or spanning gaps between contigs appear to exist. When this occurs, we will:

• Search for extending BACs by hybridization of unique end probes (overgos or PCR products) to filters of other available libraries.
• Estimate gap sizes by fiber-FISH and attempt to span small (<10 kb) gaps with PCR.
• Order and orient contigs using marker, FISH, and other information.
• For large gaps recalcitrant to these approaches, use marker sequence information and comparative genomics to define likely sequences in the gap, and then use them as hybridization probes to identify candidate BACs for sequencing and gap filling. BACs that can be placed in gaps can become new seeds for contig extension.

BAC and chromosome Assembly pipelines

[TIGRAssembler, CeleraAssembler, Pharp/Consed, CAP3]

II. Sequence release policy and submission to Genbank

The sequence release policy is conducted in the spirit of the Bermuda agreement (http://www.hugo-international.org/hugo/bermuda.htm), which states that sequence assemblies of greater than 1 Kb should be released to the public on a daily basis on a server such as a public FTP server. Finished BAC sequences will be submitted to Genbank, with or without annotations.

Refer to http://www.ncbi.nlm.nih.gov/HTGS/ for more information on HTGS Phases 0 - III.

Status	Location	Definition
Phase 0	HTG division	single-few pass reads of a single clone (not contigs).
Phase 1	HTG division	Unfinished, may be unordered, unoriented contigs, with gaps.
Phase 2	HTG division	Unfinished, ordered, oriented contigs, with or without gaps.
Phase 3	Primary division	Finished, no gaps (with or without annotations).

III. Gene Nomenclature Conventions

1. Introduction

The biological community moves towards a unified system for naming genes. Earlier sequencing projects have already defined nomenclatures that should be adopted by other model organisms in order to move towards a more common genetic language that will facilitate structural, functional and evolutionary comparisons of genes and genetic variation among organisms. Recently, the rice sequencing groups have adopted, and developed further, many of the conventions introduced by the Arabidopsis and yeast sequencing projects. This document is based on the rice project gene nomenclature document, adapted for the Solanaceae.

The rules for tomato gene nomenclature used to date can be found on the web (http://gcrec.ifas.ufl.edu/tgc/newsletters/vol42/v42p6.html) and in the TGC 20:3-5 and TGC 23:3. These documents spell out the rules for naming chromosomes, linkage groups, genes, alleles, mutants and genetic rearrangements. The conventions given in this document should largely supersede most of the conventions given in the TGC.

2. Gene naming conventions.

2.1 "Traditional" gene names based on mutants

Gene names should be given preferably in English. If a language other than English is used, the name should preferably not contain any accented or umlauted characters. The gene names can be of two types: They should either briefly describe the phenotype or relate to the gene function. Before naming a new gene, an extensive search should be conducted to ensure that it is unique. The SGN database will provide a searchable list of gene names and a facility to register one's new names. Traditional gene names accommodate alleles but usually do not capture alternatively spliced forms.

Gene names consist of the following:

2.1.1 Gene full name

The full name consists of a name and a locus designator. The name should briefly describe the salient characteristics of a mutant phenotype. The locus designator serves to differentiate genes at different loci that affect the same phenotype.

The first letter of the gene name is always capitalized with all following characters in lower case, regardless whether the first allele was dominant or recessive.

2.1.2. Gene symbol

A gene symbol consists of two parts, the gene class symbol and a locus designator. The gene class symbol consists of 3 to 5 letters (preferably 3) and should be derived from the gene full name, followed by a locus designator that serves to differentiate genes at different loci that affect the same phenotype.

The first character of the gene is always capitalized, regardless of whether the first allele discovered was dominant or recessive, and all other characters are lower case. Gene class symbols should always be written in italics, with the locus designator, which is not written in italics. Together, the gene class symbol and the locus designator form a gene symbol, which must be unique for that organism. For genes lacking the locus designator, a "1" is assumed. Locus designators are attributed in order of discovery.

The prefix Le (e.g. for tomato) is not part of the gene symbol, although it may be used in certain contexts where it is necessary to point out that the tomato gene is considered and not the gene from another species (LeCHS vs AtCHS). [It is understood that tomato genes are prefixed with Le because of historic reasons and not as an endorsement for that classification nomenclature. The official name of tomato is now Solanum lycopersicon.]

Gene names are never deleted. When a conflict occurs (such as when two loci are recognized as being allelic), one gene symbol will be retained as a synonym of the other.

2.1.3 Alleles

Different alleles at the same locus are distinguished by adding a hyphen and numerical suffix to the gene full name or gene symbol.

2.1.4. Protein full name and symbol

The protein full name should be identical to the gene full name, but in all uppercase and italics. The protein symbol is the gene symbol written in uppercase and italics.

2.2 Gene names based on function

Function based gene names (such as CHS, chalcone-synthase) are treated the same way as mutant based names in terms of gene symbols and alleles. The name refers to a chemical or enzymatic function. The abbreviation should also be 3 letters. It is recommended that the protein be assigned synonyms describing its molecular function according to IUPAC rules or an EC number (EC: 1.1.1.1).

2.2.1 Post-translational modifications

In cases where a post translational modification, such as protein splicing, leads to the formation of two or more protein molecules with different activities or functions, the spliced protein molecules will carry protein names and symbols consistent with their molecular function or associated phenotypes.

3. Systematic Locus Ids

Term definitions:
Locus: a segment of chromosomal DNA that has been identified as containing a gene or pseudogene. The locus starts at the transcription start site and ends at the transcription end site. A locus name is a 'bag' that comprises all alleles and alternatively spliced transcripts. In this definition, the promoter is not part of the locus.
Gene or Gene Model: a gene model is the structure of a transcript mapped back onto the genomic sequence and that specific sequence. It corresponds to a specific transcript, e.g., a specific splice variant.
Pseudogenes: A sequence in the genome that has gene like characteristics but cannot be transcribed or translated (e.g, due to stop codons). [A more precise definition of pseudogene would be desirable].

3.1. Nuclear genes

BAC-based names: During the sequencing project, BAC-based systematic names will be assigned for predicted or experimentally verified genes on BACs. The BAC based names consist of the BAC name and a number, separated by a dot, for example "F12H11.20". Numbering is incremented by 10 for each gene and starts at 5' end of the finished BAC sequence and is irrespective of strand. The direction of the BAC in the assembled chromosome is not taken into account.

Chromosome-based names: After pseudochromosomes are built, a chromosome-based name will be assigned to each locus. The name will consist of the following: A two letter code for the species: LE for *Lycopersicon esculentum* [Another possibility would be to use *Solanum esculentum* = SE], followed by two digits indicating the chromosome (01-12). The next letter is a "G" for RNA and protein coding genes. It is a "T" for transposons and "R" for repeats. The transposon and repeat names use their own number space. The letter can also be used to denote other parts of the gene: "P" is used to denote the promoter of the corresponding gene (Le08p00010 is the promoter of gene Le0800010).

[Another possibility would be to use the letter to designate the strand of the gene, as was done in yeast: W (Watson) for the forward strand in C (Crick) for the reverse strand].

This is followed by a five-digit number that corresponds to a sequential numbering from the top of the chromosome to the bottom independent of orientation. Initially, the numbers will be assigned in increments of 10 to leave room for further genes that may be identified later. [For repeats and transposons five digits may not be enough. The size of that number space will have to be defined at a later point.]

Examples : Le08g00010 is the first gene on the top arm of chromosome 8.
Le08r00010 is the first repeat on chromosome 8.
Le08t00010 is the first transposon on chromosome 8.

Chromosome-based names are case insensitive. Using mixed case enhances readability.

3.2 Organellar genomes

The chromosome digits are replaced by the letter MT for mitochondrial genes and PL for plastid genes (this is a small change from the way rice handles organellar chromosomes. In rice, the letters M and P are used to designate the organellar genome, but I think it is better to always have the format LECCDNNNNN, where CC always has two placeholders for identifying the chromosome, so that the core name is always 10 characters long).

3.3 Pseudogenes

Genomic sequences that have gene-like features but are not functional genes are termed pseudogenes [more explicit definition of pseudogene desirable]. These genes will be followed by .P suffix in the systematic locus name.

3.4. Transcripts

A locus can give rise to multiple gene products through alternative splicing and post transcriptional events. Different transcripts that arise through alternative splicing are designated by suffixes of the form: -1. The numbering is arbitrary and will in most cases reflect the order of discovery of alternatively spliced

products. Caution is required when assigning alternatively spliced genes. Not every irreconcilable evidence should be taken to indicate alternative splicing.

Example Le08g00010-3 is an alternatively spliced transcript of locus Le08g00010.

3.5 Proteins

Proteins have the same identifiers as the corresponding transcripts.

3.6 Genes in sequence gaps

It is suggested that a number space be reserved for 200 genes per 100kb of predicted gap (that gives room for 20 genes with a increment of 10). On un-anchored BACs, only BAC-based names will be used until the sequence gap is filled.

3.7 Genes on BAC overlaps and incomplete genes at BAC ends

A problem with the BAC-based annotation approach is that there will be incomplete genes at the BAC ends and sometimes duplicated annotations on overlaps. No adjustment to the BAC sequence (such as addition of sequence from another BAC) should be made to include a complete gene on a given BAC, so that the BAC sequences submitted to Genbank will always match the BAC sequences in the databases (this is a problem with some of the Arabidopsis BACs which were changed and now don't correspond to the physical BACs). Using phrap scores the overlap with the better sequence quality should be determined in the chromosome sequence. If sequence qualities are identical, the sequence of the 5' will be used. Genes that are split between BACs will be annotated on the separate BACs as gene fragments belonging to one gene. A 'polishing' step will be required after chromosome assembly for these genes.

4. Editing, deleting, inserting, merging and splitting

It is important that all editing operations are logged in order to be able to reconstruct the editing history. In the following, a way to handle editing operations in terms of namespace allocation, and a simple controlled vocabulary for logging the locus history, is proposed. For each editing operation, the date, the gene name, the editing term, and the other involved loci should be logged.

Example:
20031120 Le08g00010.1 MERGE Le08g00020
20031120 Le08g00020.1 MERGEDELETE

The old versions of the gene models should be kept in the database as obsoleted gene models.

4.1 Editing genes.

The locus ID remains the same even if the locus is edited and changes in structural or functional annotations occur, as long as the locus is not split or merged with neighboring loci. All editing changes should be carefully logged. Coordinate changes are logged using the vocabulary term COORDSHIFT.

4.2 Deleting genes

Genes identified by computational methods that may prove to be false positives are obsoleted instead of deleted from a database. This means that all the information about the gene is kept in the database and the identifier is never reused, as this would lead to confusion. The proposed vocabulary term is DELETE.

4.3 Inserting genes

An identifier that lies numerically between those of the neighboring genes is attributed to a new gene. SGN () will make available a tool that assigns new ids so that duplicate ids can be avoided. The proposed controlled vocabulary term is INSERT.

4.4 Splitting genes

When a locus identifier mistakenly comprises two genes, the locus is split in the following way: The new locus that retains the original functional annotation or contains the majority of the sequence retains the old locus identifier, and the other locus is assigned a new locus identifier. Two entries have two be logged: The first uses the vocabulary term SPLIT and refers to the identifier of the old gene. SPLITINSERT is the proposed term for the new entity. Note that this is different from INSERT.

Example:
20031120 Le08g00010 SPLIT Le08g00015
20031120 Le08g00015 SPLITINSERT Le08g00010

4.5 Merging genes

Previously separate genes are merged by obsoleting the locus name of the gene containing the smaller part of the sequence of the combined locus. The obsoleted locus should be retained as a secondary identfifer or synonym for the first one. Again, two entries have to be made to the history: MERGE relates to the gene whose identifier is retained, whereas MERGEDELETE pertains to the gene that is obsoleted.

Example:
20031120 Le08g00020 MERGE Le08g00030
20031120 Le08g00030 MERGEDELETE Le08g00020

5. Gene Classification

5.1 Functional annotations according to sequence similarity and expression

We define five classes of gene annotations based on sequence similarity.

1. "Known Gene".
Annotation: Report only name of the gene it is identical to. (Example: CHS).
Comment field: identical to.

2. "Strong similarity".
Strong similarity means low evalues (e < 1e-100) but some degree of judgment by the curator is required for different gene families.
Annotation: "Putative" + name of gene (Example: putative CHS)
Comment field:

3. "Similarity".
Annotation: "Similar to" + name of gene or gene family (example: aquaporin family)
Comment field: -

4. "Expressed gene"
The gene has EST/cDNA evidence.
Annotation: "Expressed gene" + other anntotation, such as "similar to aquaporin family"

5. "No evidence"
Only ab-initio data are available and no significant database matches have been found (evalue > 1e-10).
No expression data is available.
Annotation: "Hypothetical Gene".

6. Naming of Genetic Stocks involving insertion/deletion events

a delta symbol ("Δ") is used to indicate a deletion and a coule colon ("::") is used for an insertion. A transgenic line should be prefixed with TG:

7. Chromosome naming conventions (Extracted from TGC 20:3-5 and TGC 23:3)

7.1 The chromosomes are numbered according to their length measured in pachytene. Such numbers have already been applied (Barton, 1950); chromosome 1 is the longest, chromosome 12, the shortest. In addition to length, such features as positions of centromere and amount and distribution of heterochromatin serve to identify each chromosome. Short arms are symbolized by "S", long ones by "L"; thus "1S" designates the short arm of chromosome 1.

7.2. Linkage groups. Linkage groups bear the same numbers as their respective chromosomes. As soon as the arm location of a gene is known, the locus numbering shall be revised to reflect that information. The smaller arm of each chromosome is designated as the left arm, and the zero position is the distal or left end of the small arm.

IV. Guidelines for structural and functional gene annotations

The purpose of this section is to give guidelines for gene annotation in order to prevent heterogeneity in annotations. These are obviously minimal guidelines and more analyses can be run if so desired.

1. Gene Structural Annotation

Genes should be identified based on finished BAC sequences. The proposed standard pipeline is modeled after the Arabidopsis annotation pipeline from The Institute for Genome Research (TIGR) and The Arabidopsis Information Resource (TAIR). The guidelines comprise computational prediction, experimental annotation and verification through alignment of known sequences, and resolution of conflicts generated by the different methods. The annotation will be performed on a BAC by BAC basis and BAC based names will be assigned to each identified gene corresponding to our gene naming conventions standards (see section II). Chromosome-based names will be assigned after the chromosome pseudomolecule sequences are assembled.

For the computational identification of protein coding genes, based on experience with Arabidopsis, GeneMark.hmm (Borodovsky and McIninch, 1993) and Fgenesh (Solovyev et al., 1994) are expected to work best. GlimmerM (Salzberg et al., 1999) and Genscan+ (Burge and Karlin, 1997), and Eugene (pending availability for general use) are also recommended. The genefinders will calibrated for tomato codon usage. The tomato calibrations should be shared with other centers for maximum consistency. tRNA

genes will be detected using tRNAscan-SE (Lowe and Eddy, 1997). Other RNA genes, such as microRNAs and snoRNAs, will be identified through sequence similarity to known RNA genes.

For the experimental gene identification, Tomato cDNAs, ESTs and unigenes (and corresponding data from other Solanaceae species) to the tomato genome sequence using Geneseqer (Usuka et al., 2000).

2. Gene functional annotation pipeline

Several strategies should be combined to obtain the highest quality functional annotation from an automated pipeline: Identifying conserved protein domains, analyses of homology to other proteins, and protein location predictions (membrane domains, subcellular location). Wherever possible, the annotations should be converted to Gene Ontology codes for easier comparisons with the other annotated genomes.

For the protein domain identification, identification of Interpro domains using the software and domain databases available from the Interpro consortium (http://www.ebi.ac.uk/interpro) is recommended. Interpro domains are a selection of Pfam, TIGRfam, Prosite, ProDom and PRINTS domains that are grouped according to function and have defined evolutionary relationships. Each group of domains has an Interpro domain identifier in addition to the source database identifier (Pfam etc). We will use the Interpro domain identifier to convert Interpro domains into Gene Ontology annotations using the mapping file provided by Interpro. The main functional categorization will be based on these Gene Ontology terms, which will be mapped to one of the GO slim vocabularies available from the GO consortium to produce a top-level classification.

In addition, protein functional annotations will be generated through analyses of similarity in three stages: First, tomato proteins will be blasted against Arabidopsis, rice and Medicago (pending availability) proteins; Second, if there is no significant hit in the previous analysis, they will be blasted against Swiss-Prot. Third, if there is no significant hit in Swiss-Prot they will be blasted against the Genebank non-redundant dataset. The location of proteins will be determined using TMHMM (Krogh et al, 2001) for transmembrane domains, and TargetP for subcellular location. These results should also be translated into Gene Ontology codes.

3. Resolution of conflicts

Many of the predictions and the experimental gene identifications will result in conflicting evidence for a given gene. Number of exons, lengths of untranslated 5' and 3' sequences, and joined or split genes will have to be resolved to make a gene call. In some cases, the conflicting evidence will be due to alternative transcription start sites and alternative splicing. We will use PASA (Haas et al, 2003) and GeneSeqer to automatically determine the best gene model for the given data. The instances that cannot be resolved using these tools will require manual curation. The main tool for the manual curation will be the open source Apollo genome viewer (http://www.fruitfly.org/annot/apollo). Similarly, conflicting information in functional annotations will have to be resolved. For the functional annotations, the similarity-based classification will be checked for consistency with the interpro derived annotations.

V. Training and Quality Control

Cornell commits to implementing a pipeline that conforms to the above guidelines and makes the pipeline publicly available. Cornell will also provide training for project members wishing to use its pipeline. A dataset of already available BAC sequence data will be used for assessing different pipelines, quality control purposes and comparing annotations from different centers.

VI. Data format standards

1. Structural Genome Annotation

A large number of data formats have been developed over the years for bioinformatics applications. The following is a list of formats that were or are being developed for structural genome annotation: GAME XML (http://www.fruitfly.org/annot/gamexml.dtd.txt), GFF (sanger version 2, http://www.sanger.ac.uk/Software/formats/GFF/), GFF3 (http://song.sourceforge.net/gff3.shtml), Genbank flat file, Genbank ASN.1 (http://www.ncbi.nlm.nih.gov/Sitemap/Summary/asn1.html), BSML (http://www.bsml.org), Chado XML (http://gmod.org), TIGR XML (ftp://ftp.tigr.org/pub/data/a_thaliana/ath1/tigrxml.dtd) and AGAVE (http://www.animorphics.net/agave/schema/v2_3/agave.dtd).

Recommended formats:

GAME XML
GAME is an XML-based format that was developed specifically for Apollo. It covers structural and functional genome annotation and also stores meta information such as evidence. Because it is so complete and Apollo native (most bioinformatics centers will probably use Apollo), this is the most logical choice for a project wide standard to exchange genome annotation information. For a complete DTD, see http://www.fruitfly.org/annot/gamexml.dtd.txt.

GFF
The Generic Feature Format is a tab delimited format for easy parsing but has many variants. A new GFF3 version has been proposed (http://song.sourceforge.net/gff3.shtml) that retains backward compatibility and removes some of the shortcoming of GFF. The version used by Apollo is GFF Sanger version, version 2. Many other derivatives exist. The best may be to stick to the GFF Sanger version 2 standard (see http://www.sanger.ac.uk/Software/formats/GFF/) as an alternative to GAME XML.

2. Raw Sequence data

Chromatograms should be made available in scf format.
Raw sequence data should be made available in Fasta files, including phrap score files.

2. Locus history data

See section II for a description to store locus history data.

3. Functional Annotation data using GO

The GO flat file format is recommended (http://www.geneontology.org).

3. Microarray Data

MageML is recommended (http://mged.sourceforge.net/software/index.php).

VII. Guidelines for software and supported platforms

For development of new software in the tomato genome project, the following guidelines should be considered. Preference should be given to UNIX-like operating systems such as Linux, and open source database systems such as mysql and postgresql should be preferred over commercial ones.

Here is a copy of the Generic Model Organism Database (GMOD) software and hardware requirements document that can be used as a guideline for the software development in the Solanaceae Project. Note that GMOD also defines data exchange standards for which it relies on Chado. Additions to the GMOD specification: 1) The mysql database server (http://www.mysql.com) should also be included for the Solanaceae project. 2) The recommended genome viewer is Apollo (http://www.fruitfly.org/annot/apollo/).

GMOD Developer's Guide
Favored Platforms:

This list shows what a system administrator would be expected to support in terms of infrastructure to run a GMOD application. Applications that require things not on this list will be looked on with disfavor unless very self-contained, e.g. PathwayTools' use of Lisp, which comes bundled with install.
Server side:

* The following environment variables should be set:
    - GMOD_ROOT
    - CHADO_DB_NAME
    - CHADO_DB_USERNAME
    - CHADO_DB_PASSWORD
    - CHADO_DB_HOST
- CHADO_DB_PORT
* OS: any of Linux/BSD, Solaris, OSX. Windows: optional as cost/benefit less favorable.
* Web Server: all of
    - Apache
        + With Modperl 1.3, temporarily; need to migrate to Modperl 2.0 (currently broken) or CGI soon.
        + Allen, Ken to investigate effort required to port their apps to ModPerl 2.0 or CGI.
    - TomCat 4.1 (auxiliary Web server to support server-side Java/JSP)
* Languages: all of
    - Perl 5.6.1 or higher
    - Java 1.4
* DBMS: PostgreSQL 7.3
* Compilers: gcc on Solaris
* Libraries: Perl bundle; requires some C libraries
* Hardware:
    - Recommendations based on current MODs to be determined
    - Minimum: 20GB disk, 512 MB ram, 1.8 GHz Pentium or equiv. Client or server.

Client side:

* OS: any of linux, windows, OS-X
* Browser: any of Mac IE, Win IE, Netscape. Which versions within these?
    - Fly and worm will analyze server logs to determine largest market segments.
    - How to test?

Server Side Directory Structure

*

$GMOD_HOME=/usr/local/gmod/ (default -- set before installing)

bin/ -- all scripts & executable
Prefix all executables with 2-3 letter app prefix
sbin/ -- system binaries that may do dangerous things
web/
htdocs/
db
cgi-bin/
appname
webapps/
appname
conf/ -- all configuration files (attr/value format)
gmod.conf
always source this first; global site params
db.conf
one for each datasource at the installation
server, port, DB, login, password,
banner/site-specific page config info
optionally source one of these after gmod.conf
according to "db" URL param
Perl: use CPAN INI parser to load
lib/
appname or shared library
Perl plugins, jar files.
doc/
appname
log/
examples/
appname
src/
appname

* $CLASS_PATH := $GMOD_HOME/lib[/app]
* Suggested CGI Parameter Names:
    - db=short symbolic name of datasource (MOD), used to look up conf file
    - class=object class within db to search
    - id=unique primary key
    - name=human readable, possibly ambiguous string
* What if multiple DB instances running on same site?
* Package manager?
* Testing process: is a goal. "Testing is good" -- LS
    - Developer side
    - Release testing: alpha, beta
    - Installer side

Interoperation

* Support chado schema for genome features

* Establish clear naming conventions for Chado extension to avoid intermodule conflicts: Use namespaces (using PostgreSQL "create schema" or table prefixes?) corresponding to Chado modules. Or will current conventions suffice?
* Different apps should share portions of schema wherever that makes sense
* Schema changes need extensive coordination, versioning and testing across all affected apps.
* Apps are encouraged to be backward compatible with older versions of schema.
* Need to track which app versions work with which schema versions.
* Releases:
    - Major (or "public") release is a coordinated release of all apps, with testing.
    - Minor releases / tagged versions as needed by development groups, no guarantee of installable tarball.
* Schema change coordination manners: notify gmod-schema list of all proposed changes. Is this sufficient (for now?)
* Support a few standard external reps of data:
    - Sequence feature apps should read/write GenBank & EMBL formats (BioPerl and BioJava handle this). GFF? GTF? BED? GAME? BSML? TIGR add BSML adaptors to BioPerl?
    - Citations: Medline flatfile format
    - Expression: MageML for microarrays
    - Maps: use CMAP tab delimited format. (the new standard :-)
    - Adaptors for chado-DBI and/or chado-XML?
* App<->DB communication:
    - SQL (DBI/JDBC)
    - TIGR developing Perl API for chado: get central dogma model, cv_term2genes, writes
    - UCLA has developed a Perl middleware API (Class::DBI) for Chado
    - BioPerl2chado -- CJM readonly API -- out of date?
    - Chado XML Dumper
* Bulk dataloading
    - Colin's GAME2chado initializer
    - Chado XML Loader
    - GAME2chadx, chadx2game
    - BSML2chado loader @ TIGR