IN PROGRESS WORKING DOCUMENT

December 2011 Design Workshop Group Proposed AIP Requirements *(draft)*

Contents

1. Suggested guidelines for AIP development strategy

This section seeks to offer some brief guidelines on pragmatic strategies for funding, developing and operating the Arabidopsis Informatics Portal.

1) Re-use: Where feasible seek to reuse existing code, previous investment or existing infrastructure. Where possible, AIP should re-use existing and proven technology, especially in the near term, so as to reduce risks and accelerate development. This is important with such aggressive time-lines, allows leveraging of existing investment already made in systems development, and offers clear opportunities for commonality of platforms.

2) Openness: Large-scale scientific systems and the research based on them will benefit from open data, open access and open source principles. The development of AIP tools and resources should adhere to the principles of open data, open source, open standards (ODOSOS). This has several long-term benefits. It promotes collaborative development between multiple partners. It allows industry to (where internal procedures allow) participate in open source development in the spirit of pre-competitive collaboration. Open access and open data have minimal barriers to adoption and promote uptake and innovation in academia as well as small and medium size enterprises. Open source is also more scalable to large computing platforms such as cloud technologies. Close source licenses may restrict the 'number of seats' or require cumbersome negotiations to adapt to the new computing landscape.

3) Compatibility: Also termed inter-operability, ensuring software or architecture components can 'talk' to one another across scientific domain, user communities or geographic boundaries.

4) Commonality vs. Competition: Competition between compatible software tools is healthy and breeds innovation, e.g. the recent upsurge and improvement in web browsers, new features and improved performance while adhering to agreed communication protocols. For a large-scale project with tight time lines competition may be less desirable, commonality or extension of other resources may be more cost effective. In particular, funding large-scale de-novo development, where existing infrastructure and openly available code exists, may be hard to justify given economic limitations.

5) This set of principles is intended to promote an open discussion and creation of a consensus on the AIP systems development and operation principles. One key aspect is the enumeration

of one or a few options for software solution technologies. Proposed technologies or architecture(s) for AIP should be assessed in terms of benefits to the Arabidopsis community, but the applications and utility of the AIP data, applications and analysis tools to other plant science domains must clearly be taken into account. In terms of cost estimates distinct project development and operational estimates should be established. The latter will clearly have components of staffing (curators, software and database team) but a future significant component may be hardware infrastructure costs that grow as data volumes rise with the outputs from current and future population wide genome sequencing and other high throughput technologies. Possible synergy in sharing or distributing these costs with existing infrastructures (e.g. iPlant, Elixir, etc.) should be explored.

6) Selection of a technology stack should be assessed in terms of guiding principles of maximizing benefit/cost, interoperability of key scientific data with other resources and reduced duplication of effort across related (but distinct) scientific computing efforts. Close coordination with the wider plant science community at the level of standard representations of data, commonality of analysis tools would likely make the AIP more attractive to potential funders as applications in crop sciences might be made clear and explicit. As an example, the opportunity to have AIP data co-located with data for crops such as rice and maize would be something that has clear commercial benefits. This should be available to crop researchers but not force non-Arabidopsis data on the Arabidopsis community.

2. User Interface

Overview

The design of the AIP user interface (UI) is critical to the portal's successful adoption by users. The UI must be web-based and simple so as to be easily accessible by users at all levels. It should provide core functionality that manages the complexity of the portal for users and should provide help and user tips in a context-specific manner. It will be important for the AIP to have a simple, yet powerful query UI that supports query mechanisms, potentially including natural language query processing and semantic queries. However, the UI will also need to present sufficient 'layered' complexity to allow more sophisticated, or 'power', users to efficiently access and utilize full portal capabilities. To maximize utility and flexibility, the AIP should support a robust common user space while supporting individualized user space, e.g. a Dashboard, which is customizable based on the users' preferences with respect to the data and tools they wish to see. Finally, integrating social communication capabilities will serve to leverage community knowledge and widen the knowledgebase.

Requirements

The appearance and user-friendliness of the portal's UI will be arguably at least as important as the underlying infrastructure. Community buy-in is of utmost importance in the wide adoption of a new tool, database, or resource such as the AIP and will rely, in large part, on whether the resource is simultaneously easy to use, customizable, and comprehensive.

1) Individualized space, the 'Dashboard'- An initial view in the portal would be generated based on user interest (e.g., transcriptomics, pathways, GO, etc.) and each user should be able to use the version of a data object they wish (e.g., different accession or version of a reference genome. Users can select the tool they want to use and the type of query they want to perform (i.e., a single gene or a gene list.) Customizable Dashboards support personalization including the capacity to keep data private (via permissions to individuals or groups), per users' preferences.
2) Common user space- The common user interface should support data entry (e.g. data types like individual genes or gene lists, markers, polymorphisms or other molecular entities), data visualization (e.g. of multiple data types and multi-variate data); data manipulation and extraction (e.g. through use of portal analysis tools and integration between the web

application and desktop clients), and data provenance (i.e. the owner, source, and version of the data.)

3) To facilitate data entry, the UI should also allow users to upload or plug-in their own data (via programmatic/shell access) to enable their immediate integration with their information in the portal. When the data are made public and/or the procedures get mature they can be integrated into the main portal.
4) Data visualization requires portal support for a broad range of data types including, but not limited to: genome, cellular, nucleic acids, proteins, networks, subcellular, organ, whole plant, plant by environment, phylogenetic and over time. Tool development will be required to enable modeling of interactions, such as with multiple abiotic and biotic stimuli, and to display animations emphasizing four dimensions (3D across time). It is currently feasible to provide visualization in two dimensions. However, visualization frameworks that act at different scales currently do not exist for biological data. Data visualization modules are essential to enable data visualization in multiple scales. This will involve visualization of biological systems and how they change from the molecular to organismal scales while incorporating genotype and response to the environment (biotic and abiotic).
5) Analysis tools, including, for example, third-party developed applications ('apps'), need to be seamlessly integrated in the AIP and accessible through the UI. In order to guide users to good applications, a system for rating the applications and data should be implemented. To facilitate development of these applications, the AIP should also provide a toolset or support environment for developers to write the 'apps'. Personal and common data should be moveable between environments with ease, for example, between Cytoscape, WikiPathways, Apollo and the Java-based desktop genome browsers. To facilitate full integration between platforms, support for import and export to common file formats such as CSV, TXT, Excel, XGMML, SBML needs to be exposed in the UI.
6) A vital component of community standards and quality control is that the UI should display provenance of the data, ideally with a link to a summary of the full provenance. Recording this information will also allow the inference of dependency on different objects in a tool and allow notification to people using the data. This would include flagging changes by setting up RSS feeds (or the equivalent) with provenance changes. Community annotation underpins data provenance and the UI should enforce a minimal set of meta-standards for data description (see 'Community functional annotation', Topic 3.) Existing model examples can be obtained from SGN and Plant Gene Wiki, for example, and must include community-determined version, provenance, original data source and contributor (person).

Social communication and community collaboration- see more in Section 9, below

User Interface Requirements

1) The user interface should support social networking and research collaborations (e.g., the ability to link to a Facebook page for a project), and use social media to announce updates to the portal, service shutdowns or training workshops at meetings.
2) The user interface should support teaching, training and discoverability. This can be accomplished by mechanisms to guide the user through the system, and could include, but is not limited to, video-based tutorials or other social aspects to discover new functionality (e.g. the AIP could provide a YouTube channel.) Examples of additional approaches that could be incorporated into the AIP: nomination of pages and interest groups that a user may be interested in, e.g. Faculty of 1000; providing "intelligent" feedback to users such as (i) listing that users who viewed data object X also viewed data object Y and (ii) recommendations of other tools related to a user's analysis for consideration. This approach would be similar to the personalized shopping items suggestions provided by commercial sites like Amazon and suggestions would be learned by the system based on user preferences and behavior. Several well-known and widely used social media outlets for AIP UI support include Facebook,

LinkedIn, Twitter, Google+. The AIP could also facilitate conventional media distribution such as hosting YouTube video workshops or posting PowerPoint presentations. This would allow users to make broader impacts with their research, including through reaching populations that traditionally have less access to these scientific resources such as K-12 students, researchers at small colleges and universities, and the general public.

3. Programmer Interface

1. ReSTful APIs (or other web-based methods); However, HTTP as a protocol may not be appropriate for all data types.
2. The API should comply with defined standards and ontologies (owned by the AIP).
3. Should allow for Normal Users that can script, i.e. scalable API that provides for minnow users to power users.
4. The API should provide facilities to integrate existing tools.
5. Additional services listed under (B)- API

(A) Interface for programming bioinformaticians and biologists: In this context we take the meaning of "programmer interface" to mean the facilities provided for programming bioinformaticians and biologists rather than the more narrow definition of Application Programming Interface (API) (though APIs are an important and essential aspect of the facilities proposed by AIP, see below). We estimate that although these users represent a minority of users (but perhaps 25% of users in laboratories dealing with high throughput data) the number is likely to increase dramatically during the life span of the AIP. Additionally, scientific programmers are often employed in multi-disciplinary teams and enable the science across numerous projects and in association with numerous "wet bench" colleagues. The needs of programmers are therefore a critical aspect AIP.

AIP should therefore allow complete and open access to all AIP data resources. Data are stored in a range of formats, for example, in relational databases, directories of files, HDF5 format etc. The developers of each module decide upon the optimum storage mechanism for their data sets, document the storage scheme (schemas, organization of files in directories) and make it available to AIP module developers through discovery services and robust documentation. Additionally they would provide Application Programming Interfaces (API); these can be encoded in multiple layers. For example, with relational databases there might be an API built with stored routines and an additional layer built with web-services (SOAP now seems unpopular, REST seems to be favored). Programmers would have the opportunity of choosing the level most appropriate for coding their application or performing analysis.

(B) Application Programming Interface (API): APIs are used by application developers, for ad hoc queries and for building pipelines. Some applications are coded to operate through the AIP web user interface and as such constitute the core functionality for the majority of users. Users also have the opportunity of uploading their own data ("My Data") into a private read-write space on the server and have read access to all the data and the ability to call all API functions. This structure allows users to query and join their own data in the context of the whole. It also allows developers of new modules to prototype and develop new applications in the context of the core. Given that all members of the community can have an account of the server, this model stimulates innovative approaches to analysis and visualization and would put in place the infrastructure on which "visionary applications" could be built as the portal matures over time. It also lowers the bar for development because the data provided by the AIP is professionally managed by IT experts and covers the most important data-sets produced by the Arabidopsis community.

The API should provide a blanket layer that serves as a platform on which modules and novel web applications can be built. The range of services it provides should be RESTful and cohesive in their operation. While advanced developers may, for a variety of reasons, need direct access to low-level resources such as databases and data collections, the API should work at a level sufficiently abstracted that adopters need not worry about the composition of the underlying resources and instead focus on the consumption, manipulation, and mashing up of API data and applications to create novel modules with advanced functionality and broad usage.

Direct access for 'power users' (and subsequently the colleagues that they work with) allows the development of new methods, applications of the AIP, and new modules. This approach is nicely exemplified in the approach employed by the Gene Ontology (GO) database and in part the success of this system is due to open source and access by a variety of means. GO provide a web-portal (AmiGO), SQL access to the underlying database, APIs for programming languages (Perl, Java) and representations in XML and RDF (http://www.geneontology.org/GO.database.shtml).

To this end, the API should have a basic set of interfaces common across all mature web platforms as well as several AIP-specific services to address the unique needs of the bioinformatics community. This approach is common across today's cyberinfrastructures and fits in well with traditional service-oriented architectures (SoA). Because of this, we anticipate that any proposed solution would leverage other work being done in the community such as the iPlant Collaborative API, the XSEDE Gateway Developer's API, OpenStack, OGSI, etc. Such solutions already provide reference implementations used in production and can provide valuable experience and insight into the form and function of what the AIP API should look like. Common to all these solutions are the following generic services:

- Authentication and Authorization
- Identity Management
- Data Movement
- Data Management & Manipulation
- Job Execution
- Event/Messaging Support
- Service Registry
- Audit/Accounting
- Monitoring
- Information Discovery

We expect that these services will be augmented with additional services, which directly support the science use cases of the AIP and the needs of the individual modules. Examples of such services may include:

- Meta data management
- Visualization
- Publishing & Archiving
- Name resolution
- Dataset Synchronization
- SQL (for direct database querying)
- PubSub

We anticipate that the API will be one aspect of the AIP that undergoes continuous development, as it must adapt to the evolving needs of module developers. Five years ago SOAP dominated the web service world. Today the community has embraced REST. Five years from now, things will have changed again. These trends reflect not only changes in technology, but also changes in the skill sets of the developer community. The API must be flexible enough to embrace such change and allow developers to work to their strengths.

Additional considerations in the AIP API include the development of a parallel set of ontological services to semantically describe the AIP. These services would use industry standards such as RDF and OWL to describe resources in a well-supported way. The response formats of the API are also critically important. Whenever possible, community-accepted formats should be used. This could include JSDL as the description languages for jobs, OAuth as the protocol for authentication/authorization, and phyloXML to describe tree data.

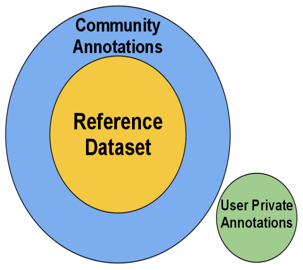
4. Curation

Overview

Curation is the activity of adding previously unknown information about an entity (e.g. gene, protein or chemical). Annotations are stored with attribution, and in some cases evidence codes and/or PMIDs. Several options exist for accomplishing curation tasks including by (a) the community as a whole in a distributed model, (b) a set of experts in the community that have taken on this role, (c) professional curators, (d) jamborees, (e) automated methods or (most likely) (f) a combination of these options. For all these options, providing a standard uniform curation interface is essential and existing tools could be adapted for this. Including Natural Language Processing (NLP) text mining tools would be helpful to facilitate curation.

Functional community annotation:

Concentric layers of annotation databases with increasing degree of AIP curation/ ownership/ trust towards the center.



Requirements

1. Query interface presents unified access point to query across layers.
2. Results can be filtered based on provenance of the information returned
3. Core Circle: AIP Reference Dataset - highest level of quality and reliability. Includes a core trusted set for various kinds of data. For inclusion here, data must conform to a prescribed schema and expert committees overseeing each data type may impose other criteria.
    a. The reference dataset can include different types of data describing molecular entities. It will provide a default view for that data type and includes (but is not limited to):
        i. Genes/other genomic features
        ii. Alternatively spliced variants
        iii. Transcriptomes
        iv. Proteomes
        v. Metabolites
        vi. Germplasms
        vii. Chromatin modifications
        viii. Genomes (natural variation)
        ix. Interaction types (protein-protein/protein-DNA and others)
        x. Subcellular localization
    b. Map associations between classes of molecules and biological entities
        i. Account for natural variation in associations

        ii. Incorporate spatial and temporal (response/development) ontologies for mapping via metadata association
    b. Vetting: addition to a reference dataset (the inner circle) requires approval by a community-sponsored expert committee. The community would be welcome to submit data to committee for inclusion into a reference dataset and easy contact methods (email contact) will be needed to communicate with committees.
        i. MASC subcommittees could organize expert committees for data types where appropriate. Subcommittees would set the quality standards for their data type and resolve conflicts.
        ii. IAIC SAB could be responsible for organizing some, or all, committees.
        iii. Alternatively, annotation jamborees could be organized
        iv. Or, paid community experts could be designated to annotate discrete data forms and to maintain gene structures.
4. Subset within the core reference dataset: Different genome annotations (previous gold standard TAIR genome annotations)
5. Large outer circle: community generated data (not curated, no barrier to deposition other than being logged in). Requires flexibility and willingness to absorb new data types on a regular basis that will be generated in the future. These could be imported into the gold standard if the community deems it necessary and appropriate.
6. Separate circle: User uploaded data available to themselves and chosen colleagues to analyze in context of reference dataset and / or within modules

Challenges:

- Adopt/develop suitable and efficient literature text mining tools.
- Motivate the community to contribute data and annotations (both carrots and sticks may be required)

5. Core Services *(section requires editing/clarification)*

a. Intelligent agents
    i. Regularly check and identify identical files; notify user
    ii. Regularly identify keywords/phrases in public data; notify user
    iii. Automatically notify user when new data of type X is made public
    iv. Automatically process user-added data according to user's needs (e.g. image processing)
b. Gold standard datasets (see curation model above).
c. Germplasm/Stock Centers

Group 2:
Defines standards for inclusion of data
Authority on stable IDs for the community
Provides authorization and authentication

Group 3:

- The hub should provide facilities to integrate diverse data sources and provide multiple dimensions and aspects upon this data (slices) that are provided to high level components.
- The hub should provide downloads of all the data and maintain 'cool URLs' for each data item so that external module databases can easily develop 'linked data' that connects the hub data and module data mutuallyT
- The hub should seamlessly interact with other organism databases.
- The hub should support a minimum set of datatypes including genome, proteomics, transcriptomics, metabolomics, phenotypes for multiple species, relationships between

components (e.g. protein-protein interaction networks, metabolic pathways, regulatory networks, etc).

Group 4:
The best way to think about this is to ask: “If you are a module developer, what would you expect to receive from the AIP core services?”
- Generally, the module developers want the data they need to build their tool, and they need a way to find out what data are available, including the type of data (e.g., “short read alignments” or “metabolite levels) and the data formats (e.g., BAM, GFF3). They also want to know: what other modules (applications) are available and what kind of data and formats can they provide? Some examples include:
- Genome sequences of Col-0 and the newly sequenced and assembled genomes and their annotation.
- To support “My Data” functionality, modules needs to know the identity of the requester. The IAP would need to manage user identities, their roles, etc.
- In general, the hub should provide all the features described in the module interoperability and programmer interface sections and storage model.

group 5:
This is where all the databases live (data store if centralized model) and
maintaining all the APIs and interoperability standards (not really a
portal, but infrastructure)
Data management, final gold standard genome(s) annotation, link to modules
(genome browsers)

group 6:
Authority for identity management/identifier assignment
(Data) model for data interoperability between modules
Contract with module providers -defining what module providers have to do
Maintenance of core reference data (need to extend model beyond single-strain reference)
Interface centred on core reference data (one of many interfaces that exist in the Arabidopsis world, focused on different use cases)
Archiving of (images) of senesced (i.e. no longer funded) databases

Responsibility for raw data archiving devolved to existing archives where appropriate archives exist. Should the hub have responsibility for upgrading data to fit successive versions, or this this devolved to the user community?

6. Modules

Overview
The AIP infrastructure will consist of a number of more or less decentralized modules providing data and services. Modules will be an important way for the global community to interact with the AIP and extend its functionalities and usefulness to the community. Small and large projects can link to the AIP through modules thus allowing individuals and groups to share their tools and resources with the rest of the plant biology community. The sections below discuss how different modules interoperate with the portal, importing data from external databases, provide a non-exclusive list of modules, and presents two potential modules as more detailed examples.

Prerequisites for efficient modularization:

1) Entities, such as genes, alternatively spliced transcripts, proteins, ecotypes, genome versions, SNPs, stocks, etc, require a defined set of identifiers. The AIP will comprise a central authority to assign and manage standardized identifiers.

2) An Arabidopsis gene nomenclature committee needs to be constituted for resolution and approvals, principally for genes and alleles. This could be a subcommittee of MASC.
3) The description of sequences, genes, phenotypes and metadata will make extensive use of ontologies, including Gene Ontology, Plant Ontology, Sequence Ontology, Phenotype Ontology, Environment, Experiment Ontology, plant disease ontology, and Chemical Ontology. (Of these, GO, PO, chemical (ChEBI) and SO exist and are widely used).

Portal-module and inter-module communication:

1) The portal will communicate with modules through an API that will likely be based on web service technology. These APIs need to be standardized by AIP and implemented by the modules. As APIs evolve over time to address new use cases, they will need to be versioned. The portal will maintain backwards compatibility for modules that do not update the APIs to the latest version.
2) Where applicable, community approved standards such as The Metabolomics Standards Initiative (MSI) [http://msi-workgroups.sourceforge.net/], The HUPO Proteomics Standards Initiative (PSI) [http://www.psidev.info/], GO and PO annotation file formats and annotations standards must be adopted.
3) The basic AIP system will consist of the portal itself, and the module providing the reference genomes. Other modules, such as the germplasm module (or several germplasm modules), will follow a plug-in architecture (based on web services), and be funded independently of AIP.
4) Modules must incorporate the IDs and associations between entities that are outlined in the gold standard data circle.
5) Modules should provide good interchange formats to allow a seamless presentation of data, and transfer of data, to end-users. It should also allow developers to build new tools based on those exchange formats.
6) Modules hosted in AIP or at other sites ought to provide programmatic access to their data using some form of Web services. For example, a rich client application or other module should be able to query AIP or an external AIP-friendly module. Modules could live on the central hub or on a site external to the hub.
7) AIP should include existing tools as modules.
   - Modules should provide features that include versioning and provenance.
   - The AIP should allow users to build workflows using different modules.

Data import from other databases:

1) The AIP portal should utilize (but not recall) data stewarded in other resources. The comparative genomics module (described below) is thus a crucial infrastructure component as an enabler of this; the IAIC infrastructure needs to be able to (i) identify related entities (e.g. genes. synteny blocks etc.) in other species and (ii) retrieve data from those resources for inclusion in the IAIC user interfaces.
2) A number of systems (e.g. Gramene, Ensembl Plants, CoGe, Phytozome) already gather genomic data from a number of species and attempt to identify orthologous entities.

Options include:

(i) Pick a winner - one system to make relationship calls for each data type
(ii) Define an interface into which alternative orthology, synteny calls etc. could be included.
(iii) Ideally, IAIC would collaborate with other expert groups to help refine algorithms or directly curate the relationship calls. One challenge is the integration of manual (expert-validated) calls with a more generic, computerized system of relationship prediction.

Non-exclusive list of example Modules:

a. Refer to “use cases and modules” working group documentation at IAIC website (www.arabidopsisinformatics.org/wp-content/uploads/2012/01/Modules-Document_for_IAIC-DW_8Dec2011_final.pdf)
b. Genome-scale metabolic reconstruction
c. Pathway genealogy
d. Comparative genomics
e. Network visualization
f. Predictive modeling at multiple scales
g. *In silico* experimentation

Examples of new Modules:

Diversity Module

SNPs, CNV, and Structural Rearrangements within and among species
e.g. *Arabidopsis* 1001 genomes data

Background

Inexpensive genome sequencing has permitted the sequencing of not only any genome but also any number of representatives of a species. This permits capturing an unprecedented amount of genetic variation across a species. Leveraging these data to understand how genetic variation affects organismal phenotype and development is an unmet challenge. In addition, Arabidopsis and its Brassica allies are particularly well suited for understanding the relationships of genotypic variation for a diverse variety of agronomically important and closely related crops species.

1. Goals:
   a. Identify all allelic variation
   b. Correlate allelic variation with phenotypic variation
   c. Correlate allelic variation with environmental variation
   d. Correlate allelic variation with epistatic interactions/pleiotropy
   e. Identify and correlate all CNV and structural variants as above
   f. Connect diversity module to comparative genomics module and other systems biology modules
2. Challenges:
   a. Have all SNP, CNV, and structural variants described to a reference genome
   b. Have all SNP, CNV, and structural variants described to a “pan genome”
   c. Have all SNP, CNV, and structural variants described to an “ancestral genome”
   d. Rapid migration of SNP, CNV, and SV to new version of “Gold Standard” genomes
   e. Technical challenge: have SNP, CNV and SV outputs pre-computed (e.g., ENSEMBL plants) or calculated on the fly (e.g., CoGE or Nordborg GWAS tool)
   f. See text below (*SNP* and *CNV and structural rearrangements*)
3. Leverage existing systems
   a. 1001 Arabidopsis tools of Ecker, Mott, Nordborg, Weigel
   b. Ensembl Plants and Gramene (Paul Kersey and Doreen Ware)
   c. CoGE and iPlant (Eric Lyons, Haibao Tang)
   d. EvoPipes (e.g. Allele Pipe, Mike Barker and Katrina Dlugosch)
   e. Expand to all boutique plant databases (Solanaceae, legumes, etc)
   f. See text below for discussion of possible solutions.

SNPs

The absence of a repository for variation calls in plant genomes is a notable gap in the overall information infrastructure. While dbSNP serves the medical research community, it lacks the capacity to provide rapid integration for non-priority species. The EU FP7 transPLANT project has recently received limited finding to begin to address this area.

The Arabidopsis-specific challenge is in moving from an archive of SNP calls to a reference catalogue of genomic variation (to be used, for example, in online GWAS tools such as those currently in development by Magnus Nordborg's group).
One question is what should happen to variant calls when reference genomes are updated.
Three possible approaches are:

(i) do nothing
(ii) positionally map all SNP calls made on version x through onto version x+1, based on whole genome alignments
(iii) recall all SNPs from the original reads. There is a strong argument for doing this even in the absence of changes to the reference genome, to integrate data from different sources and allow statistical comparisons across all the data. The challenges is that this would require the group maintaining the module to recall the variants originally made by the experts involved in the sequencing projects, which could cause controversy (the problem being that the experts might not agree among themselves.)

An opportunity is provided in Arabidopsis by the efforts being made by the various groups collaborating on the 1001 genomes project to compare and standardize the protocols for calling variation across the accessions being sequenced in the project. It is possible that these standards could be applied to call other variants, independently sequenced, to the same standards, and that an ongoing community-based working group could be established to recommend their evolution over time. Regardless of whether variant calling is standardized (and whether variants are routinely recalled), protocols for should be published, possibly using an existing tool (e.g. Galaxy), ideally in accordance with recommendations for the publication of protocols made by the central group maintaining the AIP.

The development of interfaces for SNP (and small indel) visualization is already well advanced (e.g. Ensembl, the Nordborg online GWAS tool, etc.) and could be re-used by IAIC. Existing interfaces need to be developed to reflect phasing, which is vital information if the functional consequences of individual variants are to be understood.

CNVs and Structural Rearrangements
Similar issues apply to the calling of CNVs and structural rearrangements, only with greater complexity of data, and with greater challenges in user interface design (existing systems for the visualization and query of such data are in general less well advanced than for simpler data such as SNPs). Structural rearrangements in particular will have a significant impact on the functions performed by the core module, as the determination of these will expand the set (or graph) of reference sequences will need to be maintained for all Arabidopsis*,* on which shorter range variant features (as well as genes and any other type of positional feature) could be called.

A well-defined operating procedure will be needed to convert the output of re-sequencing projects into community-endorsed extensions to the defined reference sequence, and its subsequent representation in the interfaces of the variant-module.

Possible solutions:

1. Difference model to reference genome: Store CNVs and Structural Variants as variations against a reference genome. This will have similar challenges of remapping data to new reference genomes in the future.
2. New genome assembly: Store all the data as a different genome, organized by strain/ecotype.

Comparative Genomics

1. Goals
   a. Identify whole genome duplication events within a genome
   b. Identify orthologous and paralogous syntenic regions between genomes
   c. Characterize the evolutionary history of gene families (are gene families derived from whole genome dupication events (WGD) or small scale duplications (SSD, e.g., segmental duplication, tandem duplication, or transposition duplication)
   d. Infer ancestral genes (those genes being maintained at their ancestral loci), ancestral gene order, and ancestral genome structure
   e. Connect Comparative Genomics module to Diversity Module and other systems biology modules (e.g., impact of whole genome duplications on metabolic and co-expression networks to infer neofuntionalization, subfunctionlization, and dosage constraints of modules)
   f. Develop a Comparative Genomics module(s) for species of Arabidopsis, Brassica and other genera of Brassicaceae, and the families in Brassicales for IAIC, but be extensible to all flowering plants (indeed, iPlant to iLife!)
2. Challenges
   a. Identify genomes/genomic regions from different sources (and differing versions of genomes)
   b. On the fly v. pre-computed comparisons
   c. Visualization of syntenic comparisons
      i. Whole genome: pairwise v. multiple
      ii. genomic regions: pairwise v. multiple
   d. Rapid import of new genomic data (and see text below)
   e. Rapid extraction and export of
      iii. genes
      iv. genomic regions
   f. Rapid import into downstream analyses
      v. phylogenetics
      vi. evolutionary distance metrics
      vii. Provenance of analyses
   g. See text below
3. Leverage existing systems:
   a. 1001 Arabidopsis tools of Ecker, Mott, Nordborg, Weigel
   b. Ensembl Plants and Gramene (Paul Kersey and Doreen Ware)
      viii. CoGE (Eric Lyons, Haibao Tang)
   c. Phytozome (Dan Rokhsar)
   d. EvoPipes (e.g., DupPipes, Mike Barker)
   e. iPlant for Cyberinfrastructure
   f. Expand to all boutique plant databases (Solanaceae, legumes, etc).

7. Storage Model *(section requires editing/clarification)*

The AIP storage model is centralized and replicated. All AIP persistent functions shall use the centralized facility, and the storage infrastructure shall replicate all AIP data objects to all replicated stores transparently. The replicated storage model should have international scope with access to specific, replicated stores transparent and appropriate to network location. The physical infrastructure should be professionally managed in appropriately configured data centers with an approved disaster recovery plan and service level agreements acceptable to the AIP and its users. The storage model shall support high-performance computing operations.

An object is a collection of data that is uniquely identified and atomic (objects aren't nested). The AIP storage model is an object store that uniquely identifies each object and that supports transactional operations on multiple objects. The storage model provides an explicit versioning model and the ability to access any specific version of any object. Every object in the system shall be annotated with its creator/owner (a user id) and its provenance, including date and universal time of creation and its licensing (intellectual property contract with the AIP). Every version of an object shall be annotated with its modifier user id and its provenance. The storage model defines access privileges on each object using user ids and groups or roles. Access should minimally include public access (any user id may access the object), private access (specific user id may access the object), and collaborative access (specific groups or roles may access the object). Object access shall minimally include read and write privileges.

Each consortium module shall own and manage its own data within the storage model. Each module is responsible for content, data organization, and for managing access to the objects in the storage model. Each module shall supply all or part of the schema and additional Meta data for any public or collaborative data sufficient to allow a community member to make sense of the data. All public or collaborative data must be accessible in a format (a “dump”) that enables a user to reconstitute the data in a private storage model outside the AIP.

The AIP shall refer to existing raw data stores where it makes sense to do so (Genbank, for example) but shall integrate any data regardless of source required to support a specific use case.

Options

1) There are lots of different infrastructure options for object stores, but any choice should include object identity. It would be nice if the choice supported all the other requirements (replication, versioning, access control, and so on) off the shelf.
2) The number of replicated stores is to be determined, as is their geographical location.
3) There are various information processing aspects (plans, standards, and so on) that should probably be required haven’t been mentioned. They’ll need to be worked out.
4) There are a variety of versioning models available. The two most appropriate ones are repository versioning (for example, Subversion) and object versioning. There may be others, so this should be thoroughly investigated.
5) There are different and complex access models, but we should probably use a limited version of Linux access because of familiarity or some kind of privilege-based system like ANSI SQL privileges, with the requirement to keep it as simple as possible. Group and role management will probably need to have templates or something like that. Another option is to create higher level areas with a simple access scheme (making all production modules public, for example) to reduce the overhead of access management.
6) Module-based schemas and meta data can be represented in a huge variety of formats (database data dictionaries, UML models, ER models, text documents with graphics, XML schema, and so on). Appropriate choices should include consideration of the use cases (REST APIs that return XML or JSON based on a relevant schema language, databases accompanied by data dictionaries, meta data management systems). One choice would be to standardize publication on a specific schema representation. Another choice would be to require that any schema representation be computable (in the sense of being accessible to a program through a meta data schema or some kind of API).
7) There is a choice for each “object” that exists in some other storage model such as Genbank whether to replicate it in the AIP storage model or refer to it (“reuse” it).

Challenges

1) The international scope and national ownership concept is a huge challenge in terms of both infrastructure management and funding.
2) The acquisition and implementation of the infrastructure will require professional IT project management.

3) Managing access at low levels is very difficult, time consuming, and error prone. Keep it simple.
4) There will probably be tradeoffs between performance (network latency and i/o) and other requirements. The tradeoffs should be made explicit and in full knowledge of their implications.
5) Versioning can be simple or complex, and it may be difficult to reconcile a specific versioning scheme with specific use cases.
6) Referenced and replicated data will require pipelines, data version checks, and generally a lot of work to ensure that users have the right level of access to the right version of the data.

8. Data Object IDs and User IDs *(section requires editing/clarification)*

Data Object IDs: Certain data objects require unique identifiers allocated according to some specific scheme or guaranteed to be unique within their class: AGI identifiers (ATnGnnnnn), for example. These identifiers require specific processes outside the scope of the storage management system and core services in that they require integration with international standard schemes, module-specific schemes, or some other standard scheme. The data object id does not replace the storage model object identifier.

Options

1) Each module has the option of establishing a set of specific data object id schemes.
2) There may be a pan-genome object identification scheme that deals with allele variation, the presence or absence of variation, tandem duplicate variation, or large-scale structural variations (inversion). The scheme should be able to support different organisms.
3) An “ontology” may be the appropriate representation for data object ids. That ontology should include the appropriate synonyms, labels, and mappings required by any use case.

User IDs: A user is a trusted person that can be authenticated by the core services authentication system. AIP may require authentication to access any resource or may allow some resources to be accessed without authentication. The user object identity shall be represented in such a way that it is accessible to all systems that require the use of such identity (keep it simple). Every AIP service that accesses data with access restrictions must support user identification. Every infrastructure facility that manages data with access restrictions must support authorization of access by user id.

The AIP should provide a module that allows users to manage their identity (change passwords, replace expired certificates, and so on), possibly with additional profile information to support AIP use cases. AIP should also provide trusted user management services (“root” or “admin” or “curator” access) to enable system-based and module-based user management.
Summary: AIP must support full user authentication.

Options

1) There are various standards for user authentication (LDAP, JEE, and so on). AIP should pick one.
2) The AIP should consider certificate authentication for trusted users such as curators that have access to facilities beyond that of the ordinary user.
3) The big choice is between verifying that the user is a real person versus trusting that they are a real person based on their own assertion of that fact. Money being involved is probably the driving force here.
4) The AIP needs to choose between a model of having all use of the system authenticated versus having some access be “public” and unauthenticated. This is an ease-of-use and accessibility versus security tradeoff.

Challenges

1) Establishing trust ("registering" a user) may be arbitrarily complex. The simple model is a unique name and a password. The more complicated model is digital certificates or a combination of trusted user creation (trusted users creating other users) and physical authentication. Given the open nature of the information, this should probably lean toward the simple model.
2) Different national models of authentication may need to be accommodated if the system is to be truly international in scope.
3) Authentication is in some sense a "hard" problem. The AIP should acknowledge this and make explicit the decisions they make with respect to authentication processes for the site.

9. Social Media

The AIP can support the Arabidopsis community to create a pioneering model of an open and collaborative community, steering it into a new, transformative, online phase through which it will stimulate interactions between Arabidopsis researchers and other plant biologists and broader segments of science and the public. Incorporating new and existing social media approaches will enable users to share with peers and learn about the contents and resources of the portal and the portal itself.

Requirements

1) The user interface should support social networking and research collaborations (e.g., the ability to link to a Facebook page for a project), and use social media to announce updates to the portal, service shutdowns or training workshops at meetings.
2) The user interface should support teaching, training and discoverability. This can be accomplished by mechanisms to guide the user through the system, and could include, but is not limited to, video-based tutorials or other social aspects to discover new functionality (e.g. the AIP could provide a YouTube channel.)
3) Examples of additional approaches that could be incorporated into the AIP:
    a. Nomination of pages and interest groups that a user may be interested in, e.g. Faculty of 1000
    b. Providing "intelligent" feedback to users such as (i) listing that users who viewed data object X also viewed data object Y and (ii) recommendations of other tools related to a user's analysis for consideration. This approach would be similar to the personalized shopping items suggestions provided by commercial sites like Amazon and suggestions would be learned by the system based on user preferences and behavior.
    c. Several well-known and widely used social media outlets for AIP UI support include Facebook, LinkedIn, Twitter, Google+.
    d. The AIP could also facilitate conventional media distribution such as hosting YouTube video workshops or posting PowerPoint presentations. This would allow users to make broader impacts with their research, including through reaching populations that traditionally have less access to these scientific resources such as K-12 students, researchers at small colleges and universities, and the general public.
    e. Social media input (Facebook site/hash tag) to functional curation
    a. Ability to submit apps to AIP
    b. Ability to review/comment apps on AIP
    c. Ability to rank apps on AIP
    d. Allow feedback/suggestions/commentary on new data associations to a given object
    e. Users who are taking advantage of the "My Own Data" feature of the AIP would be able to share their data with each other, share workflows, and help guide each other through complex analyses. (And this should be in the user interface!)
    f. Allow the community to drive what is useful