ISSN 1539–2791

Volume 1 • Number 4 • 2003

Neuroinformatics

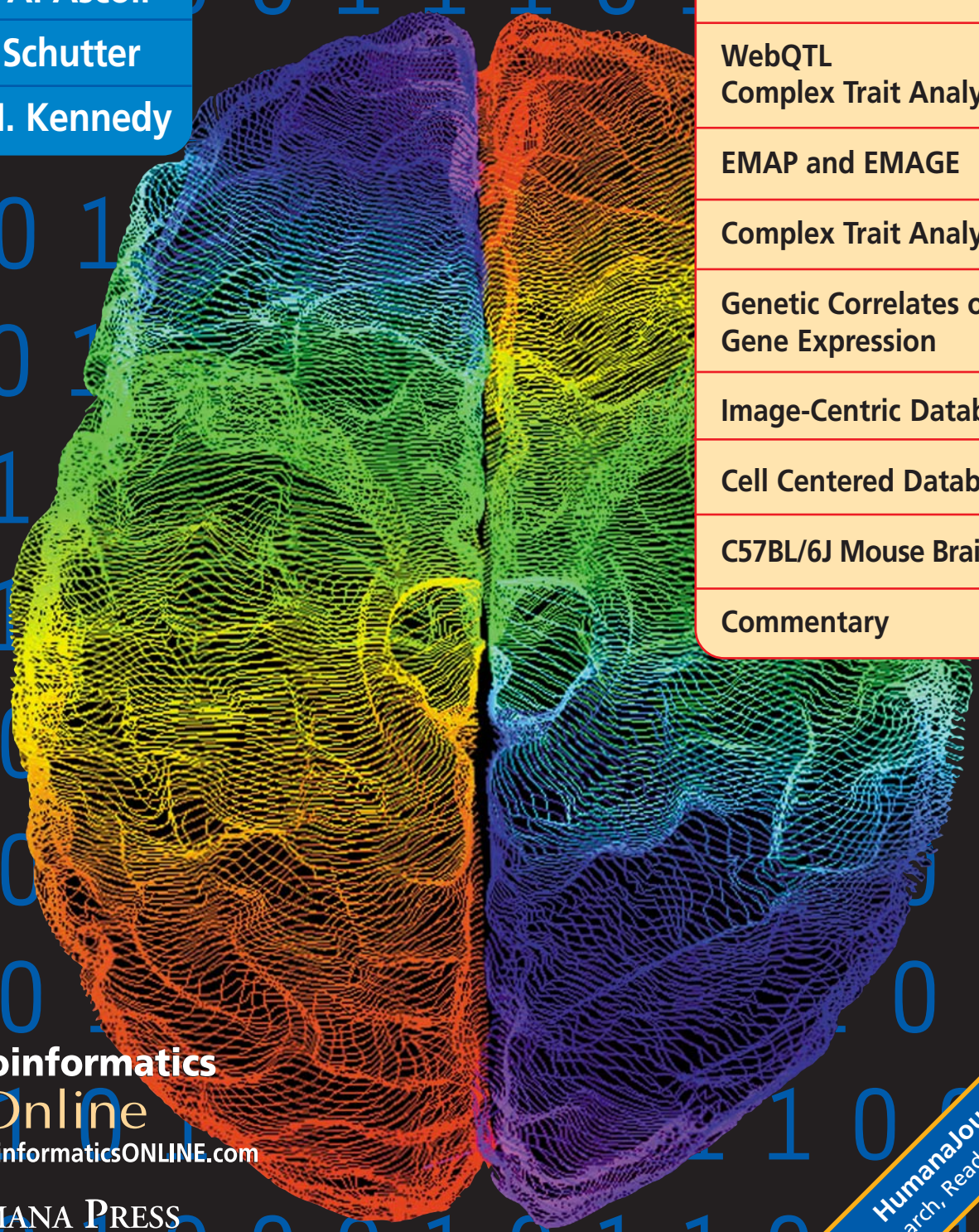
Editors

Giorgio A. Ascoli

Erik De Schutter

David N. Kennedy

IN THIS ISSUE



Neuroinformatics

ISSN 1539-2791/03/359–378/$25.00

Original Article

A Guide to Building Image-Centric Databases

*William Bug and Jonathan Nissanov**

Department of Neurobiology & Anatomy, Drexel College of Medicine, Philadelphia, PA 19129

Abstract

There is a paucity of image-centric neuroinformatics infrastructure within the individual investigator's laboratory despite the obvious need for automation and integration of experimental results. Yet, solutions can often be readily built using off-the-shelf databases and associated tools. Doing so simplifies day-to-day research operation and increases throughput. Proper construction of in-house solutions may also expedite community-wide integration of private and public data repositories. Here we describe neuroinformatics approaches at different levels of functionality, required expertise, and size of image datasets. The simplest approach offers ease of image browsing and rudimentary searching. More sophisticated systems provide powerful search capabilities, a means of tracking analysis, and even automated serial processing pipelines. In this practicum, we provide guidance in selecting among the different options.

Index Entries: Neuroinformatics; image processing; processing pipeline; computer-assisted neuroanatomy; LIMS; inter-application communication; ontology; digital asset manager; data integration; bioinformatics.

Introduction

Bioinformatics has played a critical role in fueling the revolution in genomics that has occurred over the past decade. It is inconceivable to think how that field would have progressed without the infrastructure to store, analyze, and search through the massive quantity of genomic mapping and sequencing data produced. Is there a laboratory information management system (LIMS) that would provide a similar advantage to functional genomic studies? The laboratory methods used in determining the roles of genes are diverse, and a wide range of informatics tools is needed. Public expression databases are becoming increasingly important, but most lack high-resolution and multiscale spatial information provided by standard histological methods. Such data is particularly important to investigations concerned with the nervous system where spatial heterogeneity is a central feature.

Unlike the one-dimensional data that is at the heart of genomic information, the maps produced by histological data are four-dimensional (space plus time). The attendant analytical com-

*Address to which all correspondence and reprint requests should be sent. E-mail: nissanoj@drexel.edu

plexity can lead to an image data management system being much more complex than the database systems created to support traditional bioinformatics research efforts. Furthermore, the resolution range of relevance is vast. Yet, successful methods to tame the problem are emerging. As reviewed in this issue, there are a number of public domain databases addressing this need. There is, however, a significant gap between the sophistication brought to bear by these endeavors and the solutions in place in individual investigator laboratories. Rarely do the latter make use of even the most rudimentary tools available. This is quite unfortunate, as those would greatly decrease workload and improve throughput. To aid in rectifying this, we present here a guide to implementing in-house image-centric neuroinformatics tools. Our objectives are to introduce the reader to the available options.

The needs and means of individual laboratories differ significantly across a broad range; we have taken this into consideration. We describe three levels of sophistication, beginning with a system that a small lab could readily implement without dedicated information technology (IT) personnel. This level relies on off-the-shelf multimedia asset management software and offers the advantage of greatly simplifying the tasks of organizing and viewing image files and associated information. They are called asset managers because they handle a variety of file types including sound, video, individual images, text, etc. This level 1 is suitable where only a few individuals need access and the data load is low. Even with minimal computer literacy, an investigator can easily put together a system based on this solution within days. The next level discussed makes use of desktop database solutions. With these, one can obtain much greater control and create complex customized functionality. This is much more of a full neuroinformatics solution with better database capabilities and facilities for automated control of external image processing applications running over multiple computing platforms. At this level, a computer savvy lab member is needed to implement the system and maintain it on a routine basis. The final level we cover revolves around enterprise grade relational database systems (e.g., Oracle) and an industrial strength environment to control processing pipelines. Many advantages are garnered here, not least of which is the possibility of interfacing with other bioinformatics resources. The powerful capabilities gained though come at the high cost of requiring personnel with sophisticated IT skills investing more time in developing the system and extensive regular maintenance.

The cursory description provided here presents the lay of the land. With the exception of the first level, additional training would be required to construct an image-centric neuroinformatics solution. The guidance provided here will, however, aid in deciding which level best suits your needs

Neuroinformatics Solutions

Level 1: Asst Managers

In this section, we consider a basic file management model that simplifies organization of image data dramatically. A common task faced by investigators is to routinely locate a set of images based on content, or, even more challenging, content plus auxiliary information. Current Operating System (OS) file managers are too constrained and limited in functionality to satisfy this need. One can readily improve on their notion of fixed hierarchical file organization by adding contact sheet-like viewing and searchable associated fields of descriptive information. Asset managers do exactly that. Table 1 lists commercial asset management products offering the enhancement we consider in this section.

These software applications are aimed at the multimedia/graphics market but should be

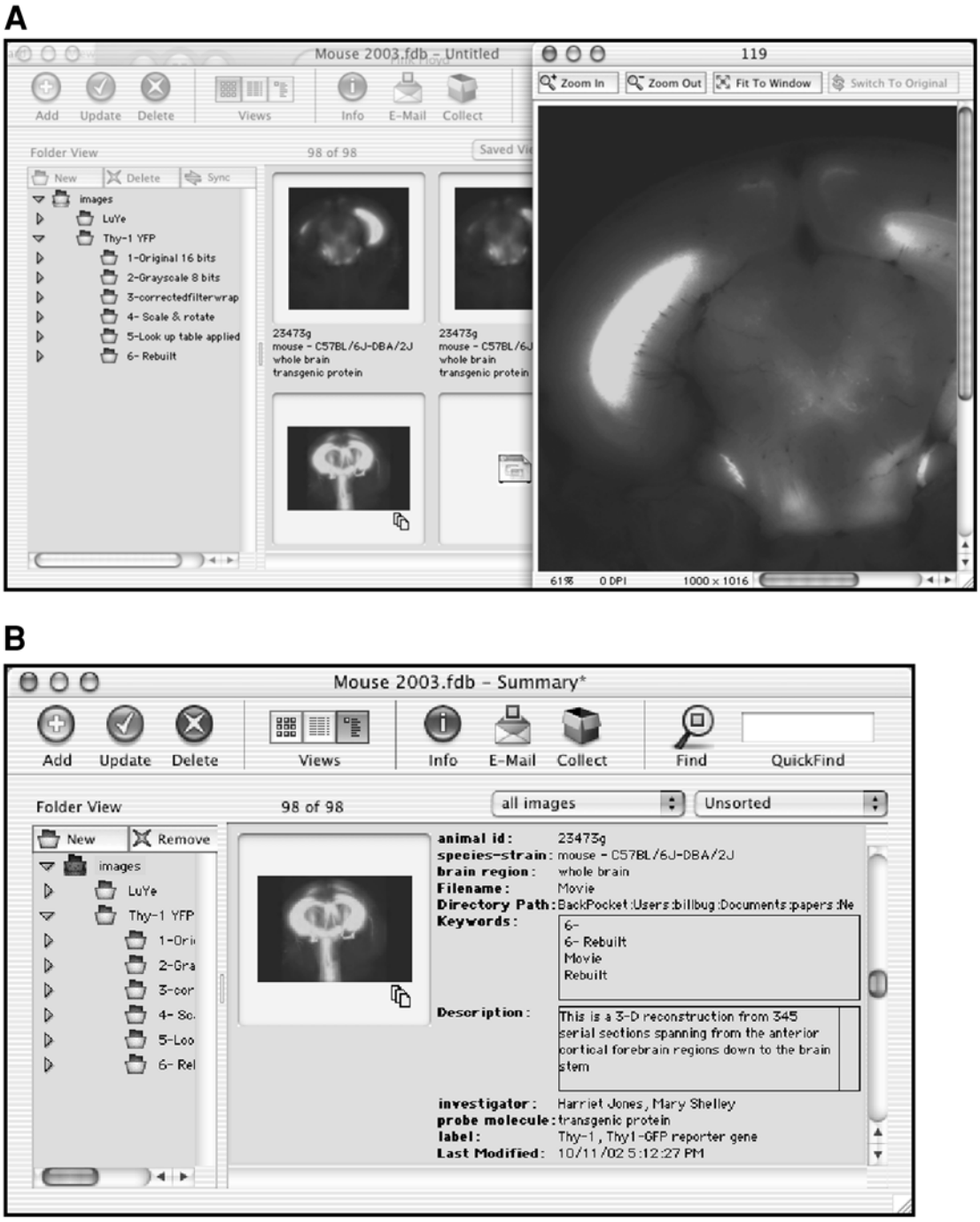


Fig. 1. Digital asset managers in neuroinformatics. **(A)** Within Extensis Portfolio, the user has access to the contents of the directories you have associated with your catalog (left panel of rear window). Directories and files can be created or deleted without leaving the program. The thumbnail view style displays small images for each item in the catalog along with custom information fields if defined (rear window). Double-clicking on a thumbnail opens the full-size image in an internal viewer (front window). **(B)** Single record mode displays a thumbnail and comprehensively lists all information on a single item. Not shown here is a list mode providing a spreadsheet-like view of the entire catalog with each item on a separate row.

Table 1
Multimedia Asset Management Software Features

Pro	*Company*	*OS*	*Fields*	*Import / File Mngmnt*	*Search / Reports*	*File Viewers/ Editors*	*Publish*	*Access Limits*	*Database connect*	*Auto*
Portfolio	Extensis, Inc.	M W	K, P, U, Srch, A, B, TD, M (EXIF)	Dir, DD; R, M, C & D; Sync	BMS, SS; T, L & R	IV, IE, T, E, P; M (~24 formats)	W, SS, CD, SVR	R, MC, MR, D, PC, CC	O, MSSQL	AS (Mac-only), B
Cumulus	Canto Software,Inc.	M W U	K, P, U, Srch, A, B, Exp,]TD, M (IPTC)	R, M, C& D	BMS, MCS, SS; T, L & R	IV, IE+, T, E; M (~130 formats); Conv	W, E, SS, SVR	R, MC, MR, D, PC, CC	ODBC	B
ThumbsPlus	Cerious Software, Inc.	W	K, P, U, B, Srch, TD, M (EXIF & IPTC)	Dir, DD; R, M, C & D; Sync; TW	BMS, T, L, R, P	IV, IE+, T, P; M (~50 formats); Conv	W, E, SS, SVR	CC	ODBC	B, DDE
IPhoto 2	Apple Computer, Inc.	M	K, P, Srch, M	Dir, DD;	BMS, T, L & R	IV, IE, T, E	W, E, SS, CD			AS
Photoshop Album	Adobe Systems, Inc.	W	K, P, U, Srch, M (EXIF)	Dir, DD	T, L, R, P	IV, IE, T, E	W,E, SS, CD	CC	—	—
Ember	Firehand Technologies, Corp.	W	P, M (EXIF)	Dir, DD, R, M, C & D; Sync; TW	BMS, T, P	IV, IE+; T; M (~24 formats); Conv	W, CD	R, MR, D, PC, CC	—	—
Paint Shop Photo Album	Jasc Software, Inc.	W	K, P, Srch, M (EXIF)	R, M & D	BMS, T	IV, IE+; T	W, E, CD, SS	CC	—	B
Picasa	LifeScape, Inc.	W	—	R, TW	T	IV, IE, E	W, E, SS, CD	CC	—	B
ACDSee	ACD Systems, Inc.	W	K, P, Srch	Dir, DD; R, M, C & D; Sync; TW	BMS, T, P	IV, IE, T; M (~75 formats)	W, SS	CC		B, DDE
CDView Pro	Tlonstruct Software	W	K, P, U, Srch	Dir, C	T	IV, IE; T, E, M (~12 formats)	CD, SS	CC	—	—

OS: M, Mac OS; W, Windows 2000/XP; U, Unix.

Fields: K, keywords; P, predefined fields; U, user-defined fields; Srch, Field data indexed and searchable; A, auto-populate fields; B, batch populate fields; TD, typed field data (e.g., text, Boolean, integer, etc.); M, interprets external metadata formats.

Import/ File Management: Dir: import at the directory level; DD, drag-n-drop import; R, rename files; M, move files; C, copy files; D, delete files; Sync, automatically reflect in asset management catalog the changes made to files in defined directories in the OS file manager; TW, importing from TWAIN imaging devices.*Search/ Reports*: BMS, Boolean multi-field searches; MCS, multi-catalog searches; SS, saved searches; T, thumbnail report; L, list report; R, record report; P, print reports.

File Viewers/Editors: IV, internal viewers; IE, internal editors ("+" indicates extensive internal editing suite); F, thumbnail views; E, link to external viewers; P, proxy thumbnail viewing of offline files; M, supported file formats; V, vector images; Conv, convert files from one format to another.

Publish: W, web; E, email; SS, slide show; CD, CD/DVD optical media; SVR, network-accessible server version.

Access Limits: R, read only; MC, modify + read catalog; MR, modify + read records; D, delete/add + modify + read records; PC, publish shared catalog; CC, create catalog.

Database Connectivity: ODBC, all databases with an ODBC driver; JDBC, all databases with a JDBC driver; O, Oracle; MSSQL, Microsoft SQL Server.

Auto(mation) : AS, AppleScript; B, batch processing within asset manager; DDE, Dynamic Data Exchange (requires programming beyond simple scripting).

Notes:

1. Based on our thorough knowledge of Portfolio, we are confident we have provided a comprehensive feature list for this product. As the features of the other asset managers were compiled from the freely published literature provided by their respective vendors, they are likely—though not guaranteed—to be complete, since these are all selling points for their products.

2. Most Asset Managers have very similar feature sets; in general, managers with a server version have the most extensive features relevant to creating shared image repositories, and for that reason we have placed those applications at the top of this list.

3. Cerius Software ThumbsPlus and Firehand Ember have a comprehensive set of internal digital image editing and manipulation features. This is not particularly useful for scientific image management, since most operations on the images are done by separate image analysis and manipulation software.

considered for use in neuroscience. Demo versions of these are freely available, and the reader is encouraged to try them. Extensis Portfolio is the tool the authors know best and is used here to exemplify how asset managers fit in the laboratory setting. Portfolio essentially implements the superset of features to be found across the other asset managers listed in Table 1.

With Extensis Portfolio, users build image catalogs by specifying directories for Portfolio to watch and import files as they are created. Alternatively, they manually import individual files or folders. It is worthwhile to invest some effort in designing naming rules for the storage directories and for the files themselves. Ultimately, an image file's unique name keeps it linked to important external ancillary data elements wherever they may appear—e.g., results spreadsheets or hard-copy lab notebook.

Visualization and file management can be done within Portfolio using one of three different modes—Thumbnail, Record, and List—as illustrated in Fig. 1.

Files, whether images or other types (e.g., spreadsheets), can be viewed in full size while thumbnails are available for navigation. The files can be grouped or sorted without impacting the actual directory location. Catalog access can be controlled using four levels of security from full control (Administrator) to read-only (Reader). Actions such as publication to a website, generation of slide shows, or burning a CD can be orchestrated on the basis of the portfolio-defined organization. Even file management operations—actions such as moving, renaming, or deleting files—can be directed from Portfolio. Essentially this software sits on top of the OS file manager. Instructions flow from Portfolio to the OS and, if required, to other applications—for example Adobe Photoshop—to edit an image.

There are many advantages to having this intervening layer between you, the end-user, and the OS. It supports the visualization described previously and provides the ability to simultaneously manipulate multiple files. Often when images are collected, one faces the quandary of whether to sort them on the basis of acquisition sessions, experimental groups, animals, and so forth. This intervening layer provides flexibility, allowing the user to have access to any and all such data organization.

Asset managers provide another valuable feature. Digital images collected for an experimental study invariably have accessory information not present in the file content itself, such as camera, microscope, magnification, fluorescent filter setting, and so forth. These essentials can be linked to an image file in a number of ways: written into a traditional paperbound lab notebook, incorporated into the file and directory names, or embedded in a hidden file header accessible only to certain programs used to view the image. Using file paths or data headers can be quite inflexible and can cause more problems than they solve. From an investigator's point of view, a paper lab notebook is often the easiest and most flexible form of storage. Asset managers such as Extensis Portfolio provide custom data fields as a more structured means of relating this data to your images, enabling you to essentially create a digital notebook. You can use these fields to search and sort your image repository. To assure data integrity, be sure to define fixed lists of terms for any given field. This is referred to as a controlled vocabulary. It avoids the obvious problems of synonyms (e.g., s. nigra, sn, s.n., etc.) and misspellings (e.g., substansia, niggra, etc.) which when used in place of substantia nigra confound your ability to sort and search.

Field entries can be made in a number of ways: manually one record at a time, in a batch mode to all selected records, and by automatically parsing header fields contained in several common image formats (e.g., EXIF format used by many consumer grade cameras).

Portfolio is not restricted to a single user. A catalog can be used as a shared resource, prob-

ably residing on a network-connected PC or Mac. This can be organized with the stand-alone version or a Portfolio Server. The latter is rather expensive and alternative approaches, as described in the following, should be considered instead. The stand-alone is designed to manage simultaneous access by multiple users and avoid conflicts with some limitations that render it of questionable value as a solution for larger research operations. Taken altogether, though, the features provided are a vast improvement over OS file management and given the very limited effort required in implementing this solution, asset managers should certainly be considered.

Level 2: Flat File Databases and Inter-Application Communication

Asset managers are designed to aid in selecting, sorting, and browsing the content of your catalog. They are not designed for extensive use with other applications. The user cannot, for example, direct processing of a batch of files by an external application based on field content. Field content cannot be readily exported or imported and you may find yourself having to reenter them manually if using multiple applications.

Scalability of the number of images managed, the amount of auxiliary information, and the number of investigators requiring access are also an issue with asset managers. If simultaneous access is regularly required by multiple lab members, and certainly if the data is to be shared beyond your local network you need an alternative solution. To achieve this greater flexibility, you need a database environment such as FileMaker Pro (Mac or Windows OS) or Microsoft Access (Windows OS). Such quasi-relational databases support these additional requirements as well as the functionality provided by asset managers. They do require, though, a bit more expertise to use effectively then do the latter. The off-the-shelf products are environments rather than turnkey solutions: they are tools for building databases, not the databases proper. They include tutorial examples with templates for data entry and reports, but you will need to build your own solution *de novo*. For a more image-centric neuroinformatics Filemaker Pro (FMP) template, see the MBL Slide Library Tutorial at (http://www.mbl.org/tutorials/mbl_tutorial/public_database.html). There are also numerous books available that can help. Our intent here is not to substitute for these but to focus on two major themes: data modeling and inter-application analysis pipelines. The first of these differentiates level 2 from level 1. Though it demands some effort, it is a key ingredient to building effective and much more extendable databases. The second theme, support for automated pipelines, is another major advantage of level 2 over level 1, and can increase analysis throughput by orders of magnitude.

In asset managers, the auxiliary information is unstructured. It is simply a list of labeled fields that are searchable. In many common settings, this is insufficient. It is often desirable to maintain a link between images from a serial processing chain, say the raw image and a filtered version. The user cannot cleanly do that with an asset manager. Calculated fields, fields whose value relies on the value of other fields, are not supported. Many investigators require this functionality. Consider the simple example of resampling an image time series. Say a series of images were captured at 1 s intervals but you wish to view them at an expedited rate. Asset managers would not support such an action.

This flexibility comes at some cost: a model accurately describing the information you are tracking must be created. It should be done explicitly as opposed to in an *ad hoc* manner. Note that this process is different from and should proceed developing the interface for entering and reporting information to the end-user.

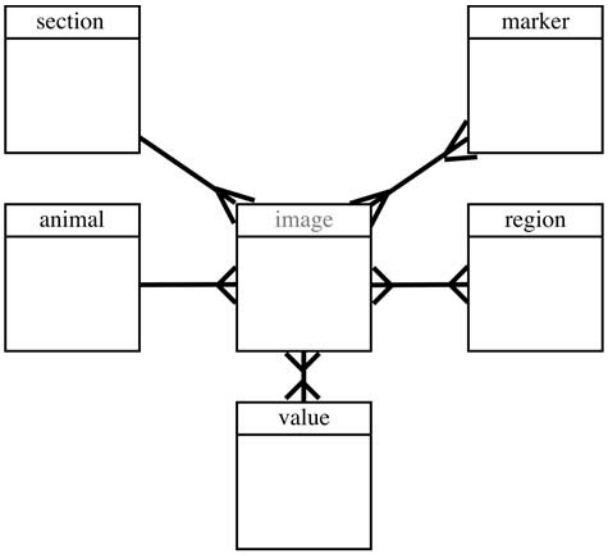
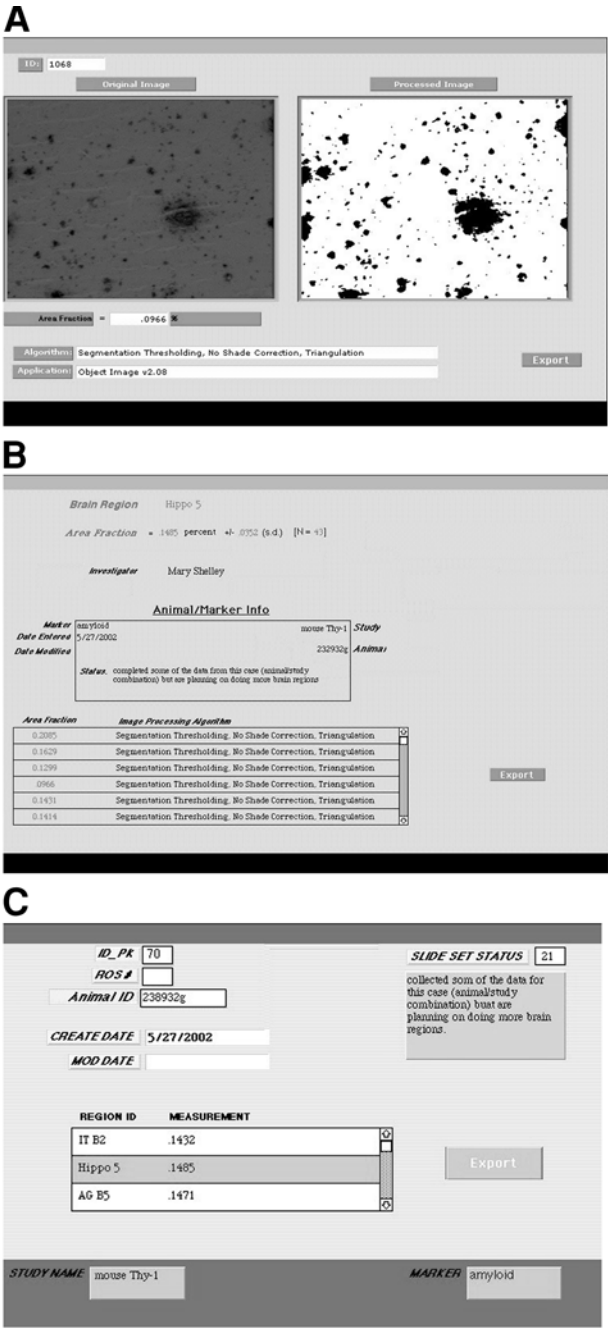


Fig. 2. Level 2 data model. Each entity contains the collection of attributes specific to its members. For example, markers have unique identifiers, the labeled moiety, if fluorescent they have an excitation wavelength, etc.; all of these attributes are specified for each entry in the markers entity. Relationships between entities have a tuple qualifier indicating how many of entity A are related to entity B. The relationship options are one-to-one, one-to-many, many-to-one, and many-to-many. Crowsfeet on the relationship link indicates "many" items from that entity are linked by that relationship. Whether the relationship is required for a given entity is specified using numerical indicators. For instance, the many-to-1 link between image and section indicates each section must be linked to one or more image(s), while each image must connect to exactly one section. In contrast, the many-to-many relationship between the image and region entities indicates each region links to zero or more image(s), while each image must link to one or more region(s).

A simple data model is shown in Fig. 2. In building a conceptual model, one breaks down the information to be stored into minimal distinct elements or entities and determines how they interrelate. A number of entities and their relations are shown in Fig. 2. Entities can represent real-world (e.g., animal) or derived objects (e.g., processing result). Entities can be further broken down into specific attributes. For example, the animal entity includes the following attributes: species, strain, and developmental stage. Think of these attributes as the columns in a spreadsheet and each row as a unique instance of that entity (i.e., a database record). Image files are, naturally, at the core of an image-centric model. Note that the images themselves are rarely embedded within the database. Instead, the image entity would contain a file attribute pointing to the location of the physical file. In building a conceptual model, one would have some entities, like those described previously, which are singular while other entities can be groupings of these individual items. There may well be information associated with a grouping that is not meaningful to the individual item. For example, images to be processed as a batch should be tracked and tagged with attributes specific to the group. In this case, processing status or results derived from collective processing would be stored as attributes of the group.

What is the relationship between the conceptual model and the actual database? In a well-conceived data model, entities and their attributes map seamlessly into database tables and fields with relationships superimposed on top of these. This results in an organized hierarchy with a database file at its root that in turn contains tables possessing individual fields and linked by relationships. This standard terminology is preserved in Microsoft Access. In FMP, entities are database files composed of individual records, attributes are fields found in each record, and relationships link files via specific fields.

Once a model is developed and implemented, data entry and presentation must be addressed. Of course there are endless possibilities here. To give a feel for those, we illustrate a solution we have built using FMP for a



large collaborative neuropathological study. It provides three distinct reports for browsing and searching as described in detail in Fig. 3.

Similar screens available for data entry are reliant on a controlled vocabulary. In multi-record views, sort order based on multiple fields can be modified as desired. Subsets of records defined by field-based filters can be exported in standard comma-separated value (CSV) format for further external analysis.

A proper data model, then, can result in a database with much greater flexibility than available with asset managers. Whether this flexibility is needed or not should be a major contributing factor when deciding whether to move from level 1 to level 2. A second consideration is whether inter-application communication is needed. Similar to what is provided off-the-shelf in Extensis Portfolio, one can manipulate files and directories within FMP or Access. One can, in fact, go much further. However, to do so does require some programming. Actions such as automatically passing information to the database from spreadsheets or text files could make use of application and system-level, interpreted scripting languages. These include AppleScript (Mac OS), Visual Basic for Applications (VBA), and Windows Scripting Host (WSH) (Windows OS). VBA and WSH are components in the newly integrated Windows development environment called "Net" which includes many other Windows-specific programming tools such as C# and JScript. Scripts are substantially easier to build than writing code in a structured, compiled language such as C++. AppleScript with its semi-English syntax is

Fig. 3. Example of a FileMaker Pro (FMP) image-centric database. Reports provide a means to review stored data. Shown here are three distinct views. **(A)** Detailed view showing all data associated with an individual image. This is similar in concept to the record view in Portfolio. **(B)** Scrollable summary of all data collected on all the images from a single tissue section and a statistical summary. Such a view cannot be generated using an asset manager. Neither could the report. **(C)** Summarizing all tissue section data associated with a specific experimental animal. The content presented in any of these reports can be set by user-defined queries on the fields' contents. The resulting view can be exported so the data can be further analyzed in an external program, such as a spreadsheet or a statistical package.

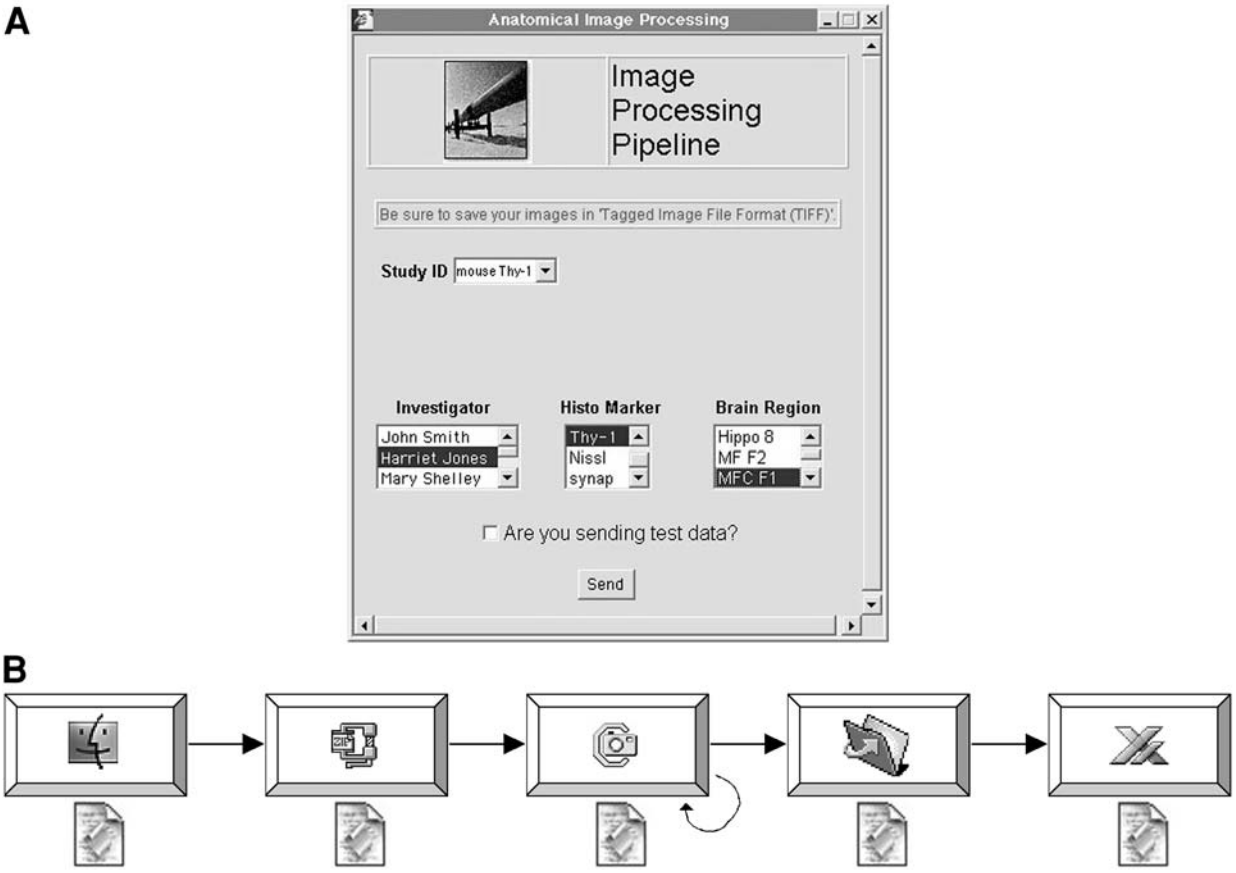


Fig. 4. Filemaker Pro (FMP)-centered processing pipeline. Completion of a data entry screen on the PC **(A)** launches the pipeline. Behind this screen is a modular collection of Windows JScripts run within a main Window Scripting Host system-level script. Users interact with this script via the Internet Explorer-like window. The JScripts create an automation layer for several Windows programs (e.g., PKZIP, FTP). When the <SEND> button is clicked, these scripts perform several tasks: they enter descriptive information related to the imaged tissue section directly into the FMP database via the Open Database Connectivity (ODBC) communication protocol; they locate the images in a specific directory on the PC and add them to a ZIP archive; they send this archive over the network via FTP to the analysis Mac. The first step on the Mac side of the pipeline **(B)** watches a specific directory in The Finder for incoming image archives. Once received, the archive is passed to a script automating ZIP decompression (MacZIP). It places the resulting set of image files in a "processing" directory. From there SCIL Image automatically performs a number of image processing procedures. The results are then automatically written into FMP and linked to the descriptive information initially added to the database from the Windows PC. Finally, all database info for this image set is dumped to an Excel spreadsheet file for further analysis. In our setting we have no need to return data from Excel back to FMP, but accommodating that would be straightforward if desired.

easy to learn. FMP and Access also contain internal scripting languages that, in the case of FMP, can be transferred unchanged from Mac to Windows, or vice versa.

A vexing problem in today's laboratory environment is the need to support multiple OS platforms. One would like to take advantage of the unique capabilities available with each in an integrated manner. Our own automated neuropathological system addresses this problem. We are using the Windows-based Stereo Investigator (www.microbrightfield.com) for unbiased stereological sampling. Images are automatically collected and passed through a series of image processing applications on Mac OS. The process is started with data entry on a PC interface (Fig. 4A) and continues on the Mac side (Fig. 4B) with images and analyzed results entered automatically into the FMP database.

Behind the PC-interface (Fig. 4A) scripts written in Microsoft JScript automatically coordinate the activity of several Windows programs (such scripts are called wrappers). This is a very common technique on the Unix platform and has been successfully employed in Unix-based neuroimaging systems such as the LONI pipeline and Fiswidgets . These tools provide a user-friendly graphical interface, reducing the level of expertise required to appropriately utilize complex chain of image processing routines. In our implementation, we adopt this technique to the Mac OS and Windows OS platforms.

On the Mac, the images are automatically processed using AppleScript to coordinate multiple applications including FMP (Fig. 4B). Once again, a master script oversees the serial activation of several wrappers by relying on a common interaction protocol. In creating a generic means for the master script to call a wrapper, we have made it very easy to add or remove a given program to the automated image-processing pipeline.

A neuroinformatics solution such as the one described here does require some time to build. The database component, including any internal application scripts, could be developed within a few months by a computer-savvy individual. System-level scripting of multiple programs across separate OS platforms producing a robust processing pipeline could be completed within a year.

Level 3: Relational Database and Robust Inter-Application Communication

The major advantages of level 2 solutions over level 1 are the flexibility offered by structured data model and the support of inter-application communication. As the scope of the neuroinformatics project grows in terms of sheer amount of data stored as well as the sophistication of the data relations, the need for better development tools and a more powerful database engine soon outstrips the capabilities of FMP or Access. The greater the flexibility and programmability demanded of the processing pipeline, the less sufficient the tools described previously will be.

The solution is to move to an enterprise-level truly relational database management systems (RDBMS), such as MySQL, Oracle, PostgreSQL, IBM DB2 or Informix Server, Sybase Adaptive Server, or Microsoft SQL Server. At the core of these systems lies a highly optimized database engine that interprets statements written in the Structured Query Language (SQL), a universal programming language for manipulating data tables. These systems are highly scalable: they support very large capacity repositories, handle rapid growth, and provide simultaneous access to a very large community of users while maintaining a high level of security. They support a wealth of programming options and standardized inter-database communication, which permits linkage of individual laboratory solutions to public databases. From the design point of view, they enable abstraction

of the data model separate from physical implementation. At this level, for a truly robust solution, one works with data modeling software—typically standalone, expensive applications requiring some effort to master. While they offer the ultimate in flexibility, RDBMS solutions require investing a great deal of money, effort, and manpower. In the absence of personnel with extensive database/programming expertise and substantial time availability, one would not attempt building level 3 solutions. Instead, one would first build a level 2 system and as this grows to it limits, invest the time of personnel with the necessary IT skill set and migrate to level 3. The recent inclusion of the MS SQL Server Engine within Access is specifically designed to encourage this type of incremental development.

In this section, we cover three aspects of level 3 neuroinformatics solutions. They are: data modeling in fully relational databases, structured knowledge, and programmatic control of internal processes and external interaction to aid in automation and integration with federated databases. The first of these is an extension of the data modeling discussion presented for level 2. Structure knowledge is, in essence, also an issue of data modeling. It is data modeling at deeper level of sophistication and with an eye toward integration with other informatics resources. Finally, level 3 automation is a great deal more powerful and its capabilities are described in brief.

Relational databases are normalized datasets. Normalization is the process by which each unique conceptual entity is abstracted and related to other entities in the data model. You have already seen a coarse-grained example of it in level 2 (Fig. 2). In a fully relational database, one would go much further and create a data model where the entities are defined at a finer level of granularity. Why is this normalization so important? One should think of a database as a computational environment rather than a static depository of information. It is very much in the same sense that we think of the sensory network within the nervous system as a computational engine rather than a passive stimulus receiver. Just as the nervous system decomposes stimuli to elemental features that are then combined to yield detectors of complex stimulus attributes, a database is a tool for decomposing information to primitives that can then be computationally combined to yield nuggets of knowledge. Breaking it down too far only complicates the process of depositing and retrieving the data; not breaking it finely enough precludes extraction of desired data features without use of sophisticated text parsing, post-processing programs. The latter is a dilemma still plaguing many efforts to extract additional knowledge from the public nucleotide sequence database, GENBANK.

We illustrate the deeper normalization process that is undertaken in a fully relational data model in Fig. 5, which further breaks down the example given for level 2.

The normalization is done as a series of discrete steps (called forms) leading to a data model or schema of greater and greater abstraction. You apply increasingly stringent criteria to remove redundancy from your data schema, creating entity relationships to share data where necessary. As in level 2, relationships between the entities are of various types (one-to-one, one-to-many, etc.) and are specified during construction of the model. These links are integral to the computational capacities of databases—they embody a significant portion of the computational rules. In addition, they are also important in enforcing data integrity assuring required attributes and entities are entered correctly.

An important advantage of enterprise level solutions is the availability of tools for data modeling and the simplicity they offer in converting model to implementation. Unlike the

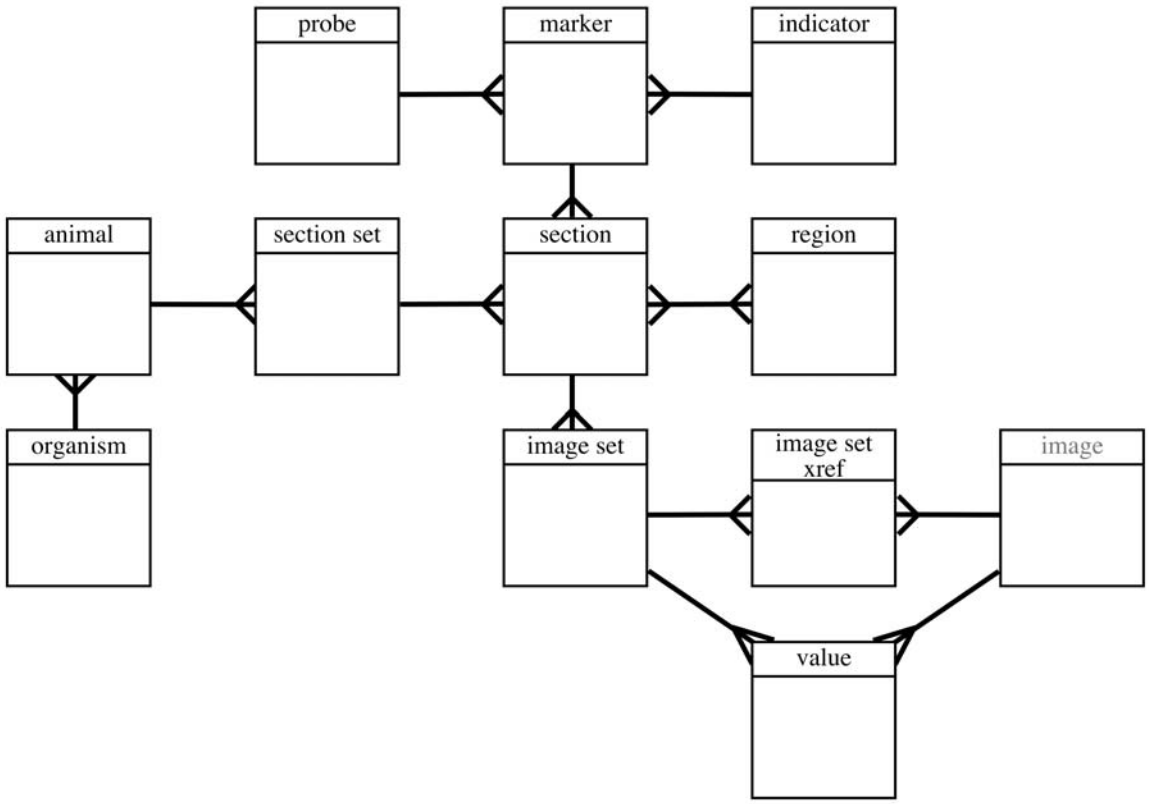


Fig. 5. Level 3 data model. These data entities and relationships model the example given in level 2 to a much deeper level of normalization. Note that each entity contains a collection of attributes specific to its members. For example, indicators have a name, either absorb or emit light, have an excitation wavelength, etc.; all of these attributes are specified for each entry in the indicator entity. Relationships between entities are represented in an identical manner to the model given in Fig. 2. There is a very significant difference, however, between the model given here and that in Fig. 2. In Fig. 2, no related entity is more than one relation removed from the core image entity. This is true because, unlike with RDBMS database engines, the FMP engine is not able to relate data from tables more distant than this. Consequently, information otherwise crammed into a single entity can now be appropriately removed to an entity of its own. This is shown here with the marker entity, which now has abstracted the probe and indicator data to separate entities. This greater level of abstraction significantly simplifies the process of re-organizing the data to meet unanticipated requirements that arise in the future and more easily supports a broad range of programmatic interactions. It can also greatly reduce the level of redundancy in the repository, which can have a significant effect on performance, as the database grows.

quasi-relational or flat file databases, most especially FMP, the data model, database tables, and the screens used to add or extract information from a RDBMS are all distinct. With a separate design and construction phase and a fully normalized data model, one has a flexible system where massive future changes can be made without requiring extensive modification of the original schema or compromising the content of the existing database.

Data modeling and database implementation requires a fair bit of IT expertise. Though

Table 2. Ontologies for Biochemistry and Molecular Biology

Ontology Name	*Reference*
BioCyc Knowledge Library	(Karp, 2002; Karp, 2002)
Biological Pathways Exchange (BioPAX)	http://www.biopax
The Gene Ontology (GO)	(Consortium, 2000)
HUGO Gene Nomenclature	(Wain, 2002)
IUPAC-IUBMB Enzyme Nomenclature (EC Numbers)	(Nomenclature, 1992; Nomenclature, 1999)
Kyoto Encyclopedia of Genes and Genomes (KEGG)	(Kanehisa, 1996)
Microarray Gene Expression Database (MGED) Ontology	(Stoeckert, 2001)
Minimum Information About a Microarray Experiment (MIAME)	(Brazma, 2001)
NCBI Organismal Taxonomy	(Wheeler, 2000)
NEWT SWISS-PROT New Taxonomy database	(Boeckmann, 2003)
PROSITE database of protein domains and families	(Falquet, 2002)

a cursory description of the process is given in figure legends 2 and 5, the details of the process are beyond the scope of this guide. While there are many books covering this topic at all levels of expertise, some of which focus on bioinformatics database design. The learning curve is steep; therefore, delving into a level 3 solution is recommended only when personnel with the requisite skill set or the time to obtain it are available.

A valuable asset of using a fully relational database environment is the availability and accessibility of structured knowledge resources. Structured knowledge is the result of utilizing a set of semantic rules and domain concepts to enable computers to automatically perform sophisticated data manipulations. Together these rules and concepts constitute an ontology. They are more than a controlled vocabulary. While controlled vocabularies are very much needed in neuroscience, given the existing conflicts in nomenclature and the problems this raises in data mining, incorporation of an ontology not only resolves this problem but also supports more powerful means of extracting knowledge from databases. At the deepest level, these ontologies allow automated inference and hypothesis generation.

A straightforward example pertinent to the present setting would be helpful. Consider an image of the CA1 region of the hippocampus in the brain captured at high-resolution. If one were to compare that image with another taken nearby, the analysis will be fraught with uncertainty if the field-of-view covered by each is unknown. Do the two images share overlapping area or not? A simple ontology could be built that would manage this type of problem. It would include concepts and rules such as magnification and its relation to fields-of-view as well as how to evaluate overlap given acquisition coordinates. This example can be extended. If the two images were acquired from different sections, the notion of alignment

would need to be added. If the images were from different animals, the notion of spatial normalization would be needed. The image processing procedures that perform these registration steps could be encapsulated as part of the rules of the ontology. Intelligent rules and domain knowledge are often combined like this in industrial robotic vision applications.

There is an extensive effort underway to build ontologies for biology (Table 2).

Unifying the nomenclature and knowledge used to describe biological systems greatly enhances our ability to share data both with other scientists as well as with computer programs. Some examples of the ontologies in neuroscience with which the reader may well be familiar include hierarchical neuroanatomical frameworks such as NeuroNames, the various tools for building model neurons and circuits such as NEURON and GENESIS, and multi-domain knowledge tools such as the SenseLab Project, the Cell-Centered Database (CCDB), and the Edinburgh Mouse Atlas Project (EMAP)/ Edinburgh Mouse Atlas Gene-Expression Database (EMAGE).

The final issue we wish to address at this level is the programmatic access possible with enterprise-level solutions. Moving data into or out of your normalized database either via automated procedures or with a manual user interface requires considerably more programming effort than with FMP or Access. Again, IT personnel are required to construct report and data entry facilities.

Associated with enterprise-level RDBMSes are industrial-strength scripting environments such as Ruby, PERL, or Python and structured programming environments such as Java or C^{++}. These provide programmatic hooks for interacting with any RDBMS conforming to SQL. These are of utility both in developing code that resides and controls processes within the database, as well as for inter-application communication. Many of these programming tools have been specifically adapted for use with biological databases, most of which are now coordinated by the Open Bioinformatics Foundation . The availability and rapid growth of such resources form a major incentive to move up to Level 3.

We are making use of these in building an Internet-accessible service that will make automated image processing algorithms available to the research community. Once images are submitted to the pipeline, analysis is performed automatically via Java-controlled image processing programs with associated data written to a RDBMS via Java Database Connectivity (JDBC). We are using the open source Enterprise Java Beans application server JBoss () to build this system and storing the associated data repository in the open source PostgreSQL (). Among the advanced features available with PostgreSQL are native geometric data types, of particular value to a digital image database.

Compared to our neuropathological pipeline described in level 2, this solution will support a much larger user-base and permits a unified interface to a disparate collection of image processing environments distributed across different platforms. While our implementation will serve a large community, the utility of this approach may also be realized even with a smaller user-base when high data throughput is required. Large-scale phenotyping efforts and drug development studies involving histological analysis of thousands of images are certainly candidates for implementing an in-house solution employing this approach.

What does it take to put all this technology into effect? Whereas a simple FMP-based data repository can be designed, built, and maintained by a single programmer, a sophisticated data management system built using an RDBMS typically involves several IT special-

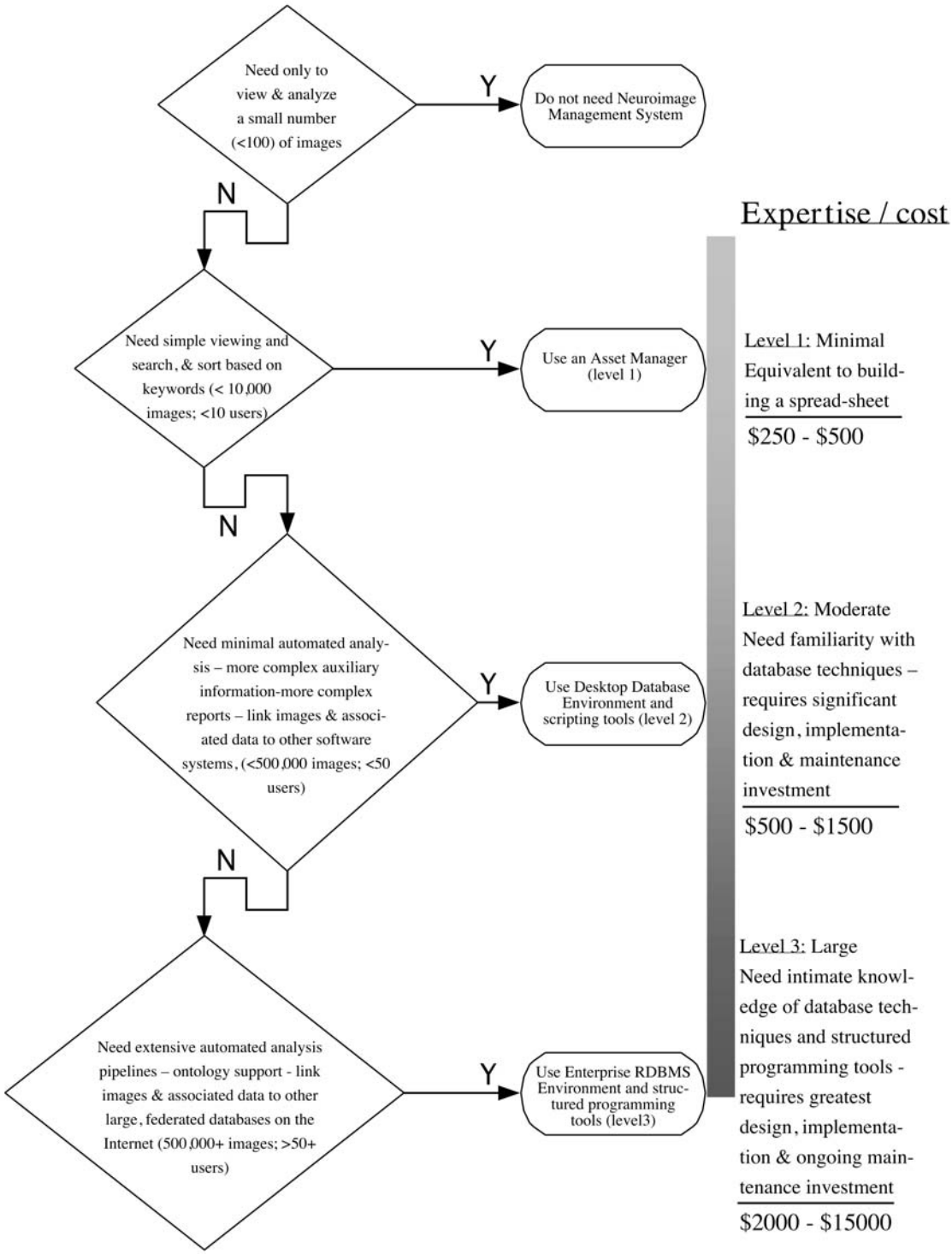


Fig. 6. Neuroinformatics infrastructure decision tree. The process of choosing an environment for managing your neuroimage research data depends on several key factors: total number of images to be accumulated, total number of users requiring access to the data—especially the number of simultaneous users, the complexity of the analysis and reporting required, the amount of automated analysis you intend to incorporate, the extent to which you need to integrate with data federations across the Internet, etc. These are the primary factors we detail in this brief practicum. One should note that these system requirements are not all-or-nothing, but rather represent a continuum of needs. Those who have amassed their own experience

ists performing a variety of roles. A database designer would design the data model and work with the database administrator (DBA) to implement it. As the name implies, the DBA would also be responsible for all administrative duties such as backup, performance tuning, security, etc. A database logic programmer would design, implement, and maintain all data-manipulation operations residing within the database, as well as constructing programs used for automatic data input and output. Finally, an interface programmer would be responsible for all the components users employ to add data to or extract data from the database. Most likely you will combine the efforts of university IT staff providing database services to the university community along with programming effort performed by in-house laboratory staff.

Conclusion

Making optimal use of limited human resources to effectively analyze large histological experiments can be daunting. As the size of the experiment grows, efficient methods to view, search, analyze, and store images and extracted results become critical. In many fields, including functional genomics, pharmacogenetics, and neuropathology, large image-centric experiments and high-throughput screening, while desirable, are unachievable without such methods. The typical image processing, visualization, and statistical analysis software used in the laboratory are not intrinsically equipped to handle large volumes of images. In this guide, we have covered how to use databases and other tools to increase productivity. The three increasing levels of sophistication described offer a wide range of options.

Your needs and available IT resources drive the decision between these options (Fig. 6). At the simplest level, off-the-shelf multimedia asset managers can liberate researchers from the onerous task of having to peruse images individually and track all associated descriptive information in spreadsheets and lab notebooks. This choice is quite adequate for laboratories whose dependence on images is low. In the midrange are desktop database environments such as FMP or Microsoft Access. Using these applications along with scripted automation, you can build processing pipelines and a web-accessible image-centric repository for up to approx 100,000 images and accessible by dozens of researchers. At the highest level of need, where many more users will generate and view a larger number of records and where complex, high-throughput image processing pipelines are employed, one should consider enterprise level RDBMS and robust programming environments.

With added flexibility and capacity comes increased cost. The level 1 solution can be implemented in a few days by most investigators and requires only minimal computer

Fig. 6. *(continued)*

implementing databases and associated data management software in a biological research lab setting may make somewhat different recommendations. With the surfeit of tools at hand, there will obviously be more than one way to create a Neuroinformatics Data Management Infrastructure of a given level of complexity. Still, along this continuum, all would likely agree when simple asset management applications would suffice, or when a full-featured, Relational Database Management System (RDBMS) is required.

(*Out-of-the-pocket costs for a level 3 system can actually be as low as $0, if one chooses an Open Source database such as MySQL or PostgreSQL and a development environment like NetBeans. However, it is very important to take into account the hidden costs incurred. The time required to design and implement an effective and robust level 3 image-centric database is non-trivial and can easily absorb 500 programmer-hours: 20 hr/wk × 25 wk, as well as requiring an ongoing investment of 2–4 hr/wk maintenance time.)

skills. A computer savvy researcher could implement level 2 over a few months to a year depending on the scope required. Level 3 requires IT personnel on an ongoing basis with working systems brought online within a year or two. Monetary outlay for database and programming environments can be rather low for level 1 and 2. A few hundred dollars will suffice for an asset manager or desktop database. For level 3, data modeling software cost about $1000 while RDBMS can range from no cost for industrial-strength open source applications (MySQL, PostgreSQL, etc.) to many thousands of dollars for commercial systems. Additional cost is incurred for computational hardware. Level 1 and 2 are implemented on desktops for image analysis and database storage. Level 2 may well require a dedicated machine for the database. For level 3, the sky is the limit. It can be implemented on a high-end desktop but is more likely to make use of a workstation.

All three systems expedite workflow within a single laboratory or a collaboratorium. Over the next decade, the technological breakthroughs needed to support seamless integration between public and in-house data storage will be achieved in many areas of biology beyond genomics and proteomics. Image-based queries reliant on computer vision algorithms and neuroanatomical ontologies will enable data mining of large, federated, image-centric neuroinformatics resources. If properly constructed, the individual database will provide a gateway for the researcher to employ these tools not only on their own repository but also across the entire federation of publicly accessible bioinformatics resources.

Acknowledgments

Supported by A Human Brain Project/ Neuroinformatics program funded jointly by the National Institute of Mental Health, National Institute on Drug Abuse, and the National Science Foundation (P20-MH 62009). Also supported by NIH grants AG017917 and MH058505.

References

Anderle, P., Duval, M., Draghici, S., et al. (2003) Gene expression databases and data mining. Biotechniques Suppl, 36–44.

Baldock, R. A., Bard, J. B. L., Burger, A., et al. (2003) EMAP and EMAGE: A framework for understanding spatially organized data. Neuroinformatics 1, 309–326.

Boeckmann, B., Bairoch, A., Apweiler, R., et al. (2003) The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. Nucleic Acids Res. 31, 365–370.

Boguski, M. S., Lowe, T. M., and Tolstoshev, C. M. (1993) dbEST—Database for 'Expressed Sequence Tags'. Nature Genetics 4, 332–333.

Bowden, D. M. and Dubach, M. F. (2003) NeuroNames 2002. Neuroinformatics 1, 43–59.

Brazma, A., Hingamp, P., Quackenbush, J., et al. (2001) Minimum Information About a Microarray Experiment (MIAME)—Toward standards for microarray data. Nature Genetics 29, 365–371.

Consortium, T. G. O. (2000) Gene ontology: tool for the unification of biology. Nature Genetics 25, 25–29.

Falquet, L., Pagni, M., Bucher, P., et al. (2002) The PROSITE database, its status in 2002. Nucleic Acids Res. 30, 235–238.

Fissell, K., Tseytlin, E., Cunningham, D., et al. (2003) Fiswidgets: A graphical computing environment for neuroimaging analysis. Neuroinformatics 1, 111–126.

Gibas, C. and Jambeck, P. (2001) Developing Bioinformatics Computer Skills, O'Reilly & Associates, Sebastopol, CA.

Hernandez M. J. (2003) Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design, Addison-Wesley, Reading, MA.

Kanehisa, M. (1996) Toward pathway engineering: a new database of genetic and molecular pathways. Science & Technology Japan 59, 34–38.

Karp, P. D., Riley, M., Paley, S., and Pellegrini-Toole, A. (2002) The MetaCyc database. Nucleic Acids Res. 30, 59–62.

Karp, P. D., Riley, M., Saier, M., Paulsen, I.T., Paley, S., and A. Pellegrini-Toole (2002) The Ecocyc database. Nucleic Acids Res. 30, 56–59.

Lacroix, Z. and Critchlow, T. (2003) Bioinformatics: Managing Scientific Data, Morgan Kaufmann, San Francisco, CA.

MacKenzie-Graham, A., Jones, E.S., Shattuck, D.W., Dinov, I., Bota, M., and Toga A. W. (2003) The informatics of a C57BL/6 mouse brain atlas. Neuroinformatics 1, 397–410

Martone, M. E., Zhang, S., Gupta, A., et al. (2003) The cell-centered database: a database for multiscale structural and protein localization data from light and electron microscopy. Neuroinformatics 1, 379–396

Nomenclature Committee of the International Union of Biochemistry and Molecular Biology and (NC-IUBMB) (1992) Enzyme Nomenclature, Academic Press, San Diego, California.

Nomenclature Committee of the International Union of Biochemistry and Molecular Biology and (NC-IUBMB) (1999) Enzyme supplement 5. Eur. J. Biochem. 264, 610–650.

Rosen, G. D., La Porte, N.T., Diechtiareff, B., et al. (2003) Informatics center for mouse genomics: the dissection of complex traits of the nervous system. Neuroinformatics 1, 327–342.

Stoeckert, C. J., Jr., Causton, H.C., and Ball, C. A. (2001) Microarray databases: standards and ontologies. Nature Genetics 32, 469–473.

Wain, H. M., Lovering, R. C., Bruford, E. A., Lush M. J., Wright, M. W., and Povey, S. (2002) Guidelines for human gene nomenclature. Genomics 79, 464–470.

Wheeler, D. L., Chappey, C., Lash, A. E., et al. (2000) Database resources of the national center for biotechnology information. Nucleic Acids Res. 28, 10–14.