Splicing UNIX into a Genome Mapping Laboratory

Lincoln Stein, Andre Marquis, Ert Dredge, Mary Pat Reeve,
Mark Daly, Steve Rozen, Nathan Goodman
Whitehead Institute for Biomedical Research
One Kendall Square
Cambridge, MA 02139

Abstract

The Whitehead Institute/MIT Center for Genome Research is responsible for a number of large genome mapping efforts, the scale of which create problems of data and workflow management that dictate reliance on computer support. Two years ago, when we started to design the informatics support for the laboratory, we realized that the fluid and ever-changing nature of the experimental protocols precluded any effort to create a single monolithic piece of software. Instead we designed a system that relied on multiple distributed data analysis and processing tools knit together by a centralized database. The obvious choice of operating systems was UNIX. In order to make this choice palatable to the laboratory biologists—who rightly consider it their job to do experiments rather than to interact with computers, and who have come to expect all software to be as intuitive and responsive as the Apple Macintoshes on their desks—we designed a system that runs automatically and essentially invisibly. Whenever it is necessary for the informatics system to interact with a member of the laboratory we have carefully chosen a user interface paradigm that best balances the user's expectations against the system's capabilities. When possible we have chosen to adapt familiar software to our user interface needs rather than to write user interfaces from scratch. We've managed to hide the power of UNIX behind the innocuous personal computer-based front ends our users know and love, using techniques that should be applicable in other environments as well.

1. Introduction

The Whitehead Institute/MIT Center for Genome Research (WI/MIT CGR) carries out large-scale genome mapping projects. A genome map is composed of a large number of short DNA sequences called "markers" which have been ordered and assigned to unique positions on chromosomes [National Research Council, 1988]. The availability of such maps greatly simplifies the task of identifying and isolating genes relevant to the understanding of development and disease. There are two main genome mapping projects at the WI/MIT CGR: the creation of a genetic map of the mouse [Dietrich *et al.* 1992], and the creation of a physical map of the human [Green and Olson, 1990]. We estimate that these projects will require the completion of several million individual experimental steps. This paper describes the design of the WI/MIT CGR informatics system and the lessons we learned during the process.

1.2 Choosing UNIX

Managing information flow in laboratory projects of this scale presents several challenges. The first challenge is managing the laboratory data for each project. The second is data analysis. The third is managing the dissemination of information both within and outside the laboratory. In addition there is a meta requirement: biomedical research protocols are a moving target. Laboratory techniques are constantly improving, and major and minor adjustments of the experimental protocols occur on a regular basis.

We chose to base our informatics system on the UNIX operating system for several reasons. First, a large number of UNIX utilities for analyzing molecular biology data already exist in the public and commercial domains. Second, UNIX is an open system that is available on many different platforms and is familiar to the academic world. Finally, the tool-based philosophy of UNIX [Kernighan and Plauger, 1978], with its emphasis on inter-process communication, lends itself to a modular design. We felt that the use of multiple modular data analysis and processing tools instead of a single monolithic piece of software would allow us to respond more promptly to changes in the laboratory protocol.

However, the choice of UNIX, rather than a PC-based operating system such as MS-DOS, OS/2 or the Macintosh OS presented user interface problems. The laboratory scientists are familiar with personal computers, primarily the Apple Macintosh, and expect software to behave as it would on a desktop system. We could not reasonably expect biologists in the laboratory to master a series of data analysis tools running under an unfamiliar operating system.

1.3. The Laboratory Protocol

A flowchart of the mouse genetic mapping protocol is shown in Figure 1. The aim of the protocol is to obtain small random DNA sequences that contain simple sequence repeats, such as $(CA)^n$, flanked on both sides by nonrepetitive sequences. These sequences are useful because they are frequently "polymorphic" between inbred mouse strains. In other words, the length n of the repeat varies between two or more strains of mice. Like eye or hair color, the length of the repeat is a genetic trait that is transmitted from one mouse to its progeny, and like other genetic traits, it can be mapped to a particular position by performing a series of genetic linkage analysis experiments.

The protocol begins by creating a library of mouse DNA sequences. This is done by cutting up whole mouse DNA into small pieces and inserting them into a self-replicating virus. Each viral clone in

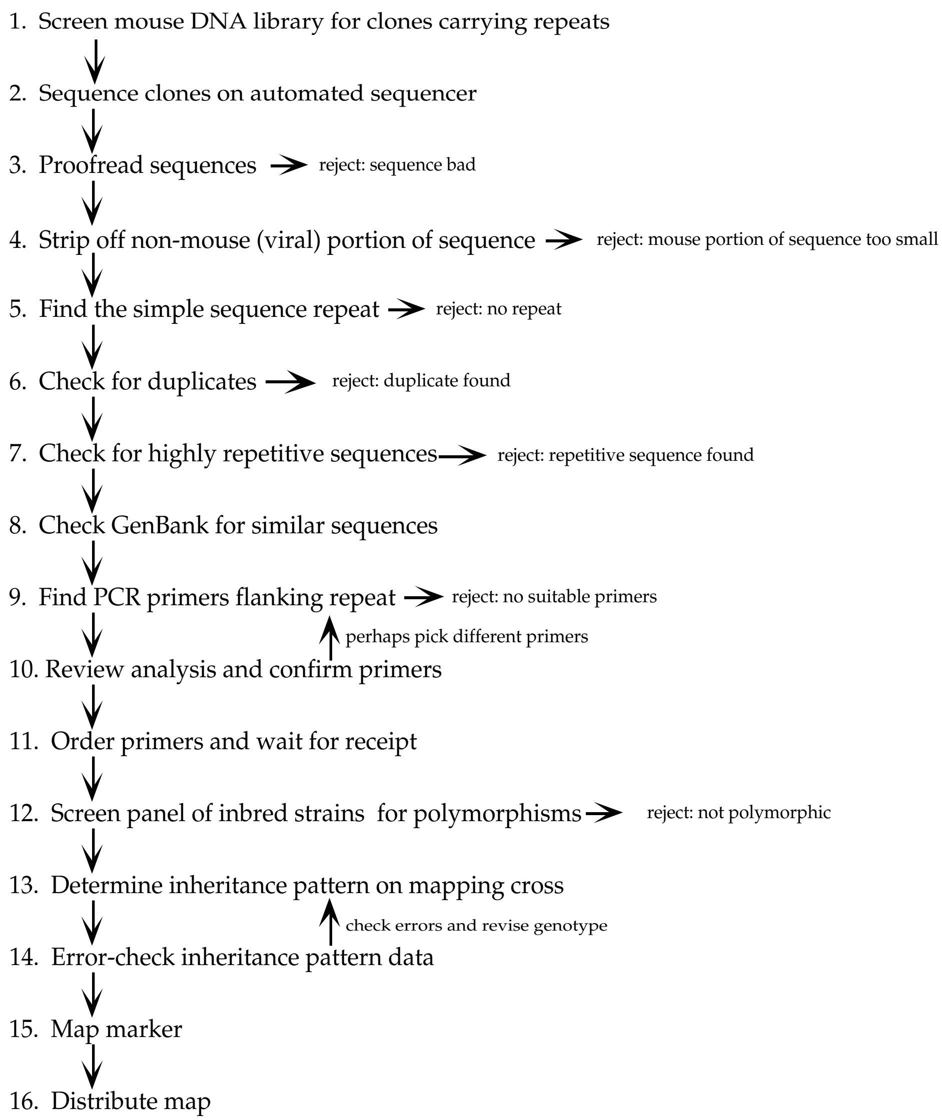

Figure 1: Genetic Mapping Protocol

the library contains a different random piece of mouse DNA. These clones are individually tested to find those that are likely to contain repeats, and each likely candidate is sequenced on an automated sequencing machine. Next comes a series of sequence analysis steps. First, since each clone consists of a mixture of viral and mouse DNA, the known viral portion of the sequence must be found and conceptually stripped off. Next, the sequence is scanned to identify the simple sequence repeat, if any. After this, the sequence is checked for duplicate sequences already present in our database and for the presence of highly repeated sequences that are present multiple times in the genome. If the sequence survives these tests, it is next compared to all entries in the GenBank database of published sequences in order to determine whether this sequence has ever been seen before. Finally, we choose a pair of short (about 20 base pair) sequences on either side of the simple sequence repeat to serve as "primers" for a biochemical technique known as the polymerase chain reaction (PCR). PCR allows us to rapidly determine the length of the simple sequence repeat without tediously recloning and sequencing it.

After having a biotechnology supply company synthesize the primer pairs, we characterize the simple sequence repeat further. Using PCR we determine the lengths of the simple sequence repeat in DNA taken from 12 common inbred mouse strains. Those repeats that are not polymorphic, that is, that do not vary in size between strains, are discarded. Those that are polymorphic are subjected to genetic mapping experiments that determine the length of each of the repeats in the offspring produced by mating two of the inbred strains. Repeats that are close together on a chromosome will tend to remain together when inherited – they will appear to be "linked" – while those that are further apart or on different chromosomes entirely will be inherited independently of each other. The genetic mapping data is now fed into a program which does the number crunching necessary to order each of the markers and determine the distance between them.

The genetic mapping protocol is essentially a data pipeline. At any given time approximately 600 sequences are in the midst of processing. Experiments can fail and need to be redone, or sequences can be found to be unsuitable and be dropped from the pipeline at various steps. While managing the protocol certainly requires ingenuity in data modeling and data processing, we have found one of the most challenging tasks to be integrating the Apple Macintosh-based work habits of the laboratory with our tool-based philosophy of UNIX.

2. Informatics System Architecture

The architecture we have chosen is diagrammed in Figure 2. The major features are:

- A centralized database running on a UNIX workstation that stores the experimental results of all steps in the mapping protocol. Our database is called "MapBase" [Goodman *et al.* 1993, Goodman 1994] and is an object-oriented database written in C++. It is a multiconnection client/server database that

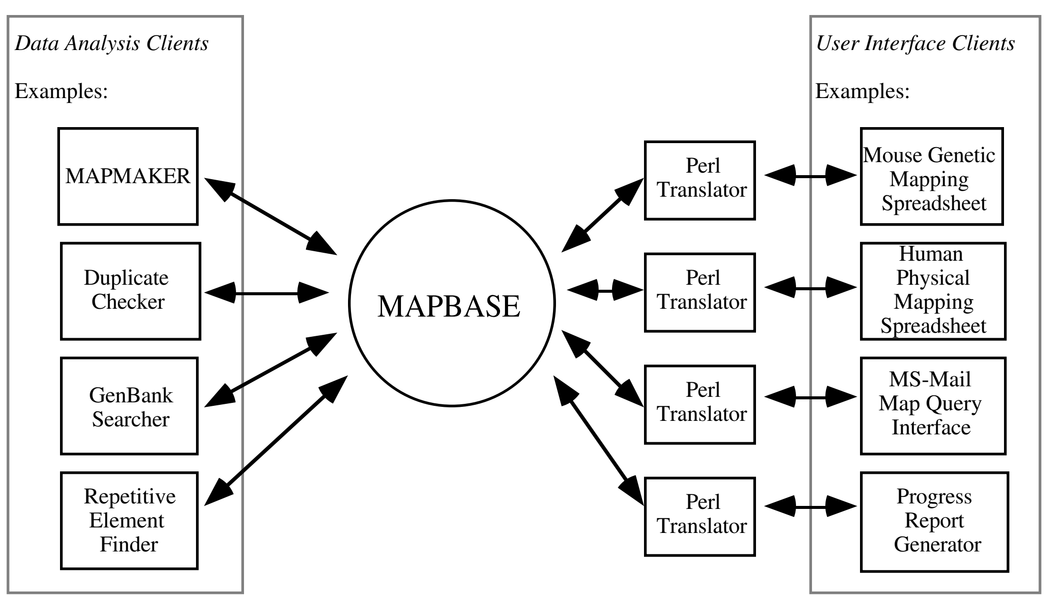

Figure 2: Informatics system architecture

supports a simple query and update language and is accessible over the TCP/IP network. An important design decision was to make MapBase accessible only to software tools and to programmers. Our end users—scientists and laboratory personnel—never interact with the database directly.

- Data analysis and manipulation clients that run automatically on UNIX workstations, often under the UNIX *cron* utility. The system is decoupled so that the database and data analysis clients do not have to reside on the same computer. Scientists do not need to take any action to initiate the data processing, but the system keeps them apprised of experimental progress by E-mailing status reports.

- Whenever human interaction with the informatics system is necessary, it is done through familiar spreadsheet, E-mail, and word-processing software running on the same personal computer the laboratory personnel use for other aspects of their work, the Apple Macintosh. This off-the-shelf software provides scientists with comfortable interface paradigms for interacting with the informatics system. By carefully matching the paradigms to the tasks, we have been able to match user expectations with the system's capabilities.

- The MapBase database uses a text-only query and transaction language which was designed to be easily machine-parseable. While many of our data manipulation clients interact directly with MapBase, most of them, particularly the adapted Macintosh programs, communicate via intermediary perl scripts [Wall and Schwartz, 1991] which handle the translation between MapBase and the client.

2.1 Choice of User Interface Paradigms - Lessons We Have Learned

Our first pass at creating user interfaces between the laboratory and the informatics system was disastrous. Adopting the conventional approach, we wrote a number of specialized applications for the graphical input and display of laboratory data. To our chagrin, biologists stubbornly kept their primary data in various Excel spreadsheets that they had created themselves, transferring the data to the informatics system in large batches only when absolutely necessary. They requested that we make our graphical data entry forms more spreadsheet-like, and were unhappy when we were unable to reproduce the full functionality of Excel. They tired of waiting for our interactive database queries to complete and developed a habit of E-mailing their data requests to the programmers.

Our second attempt to create user interfaces took a different approach. Rather than build user interfaces from scratch, we took the software and data management techniques the lab members were already using and modified them to work with the informatics system. In contrast to the first attempt, these interfaces won immediate acceptance and are still in use today. What follows are specific examples of laboratory user interfaces we have created and the general lessons we have drawn from this experience.

2.1.1 The best user interface is no interface at all

From the laboratory's point of view, computers are at their best when they work invisibly behind the scenes. When possible we have made our data processing steps invisible and automatic. For example, every new DNA sequence that is entered into MapBase needs to go through a series of software checks and characterizations: the sequence has to be checked against other sequences in the database to catch duplicates. If it passes this test, the sequence is checked against worldwide DNA sequence databases to see if it has already been mapped. Next the sequence is examined for the presence of repetitive elements (sub-sequences known to be present multiple times in the genome), and so on. Rather than ask a member of the laboratory or (heaven forbid) a programmer to initiate these tasks, the informatics system does it automatically. Every night a *cron* job queries MapBase for new DNA sequences, and feeds any that are found through a series of small programs that perform each of the characterization steps. These programs are, in fact, implemented as a series of filters connected together through UNIX pipes. The program at the very end of the pipeline gathers up the results, feeds them back into the database, and E-mails a status report to the scientists in the laboratory.

The sequence characterization programs are a nice illustration of the UNIX toolkit approach. The input for each of the programs is a series of keyword/value attributes in the format KEYWORD=VALUE. Keywords identify the sequence and the information collected on it in previous steps. Programs in the pipeline extract the attributes in which they are interested, pass through the rest, and add any information that they wish to contribute. For example, the program that determines the start and length of the simple sequence repeat looks for the following attributes in the data stream

```
NAME=MJ100
SEQUENCE=GATTGACGAGATCACAGTTTGGCACAC
ACACACACACACCAAGTTGAATTTCCTGG
```

and adds to it the following (passing through the name, sequence and any other attributes present):

```
REPEAT_START=22
REPEAT_LENGTH=20
```

The particular order in which sequence processing steps are performed is determined by a shell script, which makes it easy to rearrange the order in which processing is performed, insert data processing modules, or experiment with alternate algorithms.

Another example of invisible processing is the program that assembles genetic maps. Every night new mapping data is incorporated into the growing genome maps by a *cron*-invoked shell script and the newly constructed maps are then E-mailed to the scientists. In UNIX style, the data processing steps are performed by separate programs. A large C program called MAPMAKER [Lander *et al.* 1987], does the actual multipoint genetic linkage analysis. A second program interprets the numeric output from MAPMAKER and converts it into graphical maps, while a third converts the maps into Macintosh PICT documents and E-mails them to the scientists.

2.1.2 For moving large amounts of data use "drop folders"

There are times when it is necessary for data to be fed into the informatics system in large chunks. One example of this is the entry of raw DNA sequence information into the database. Automated DNA sequencing machines produce this data in the form of Macintosh text files. The challenge is getting these files from the Macintosh into MapBase. Rather than asking laboratory members to cut and paste these files into a specialized data entry program or to use a UNIX utility such as ftp, we have designated a "drop folder" on a disk that is cross-mounted between the UNIX and Macintosh systems. This disk appears as a NFS volume to the workstations and as an Appleshare volume on the Macintosh desktops. To transfer the sequence files to the UNIX system the user just drags them into the drop folder. A *cron*-launched perl script checks this folder periodically for new files, reformats them, and feeds them into MapBase.

2.1.3 To represent tabular data use a spreadsheet.

There are situations that require more give and take

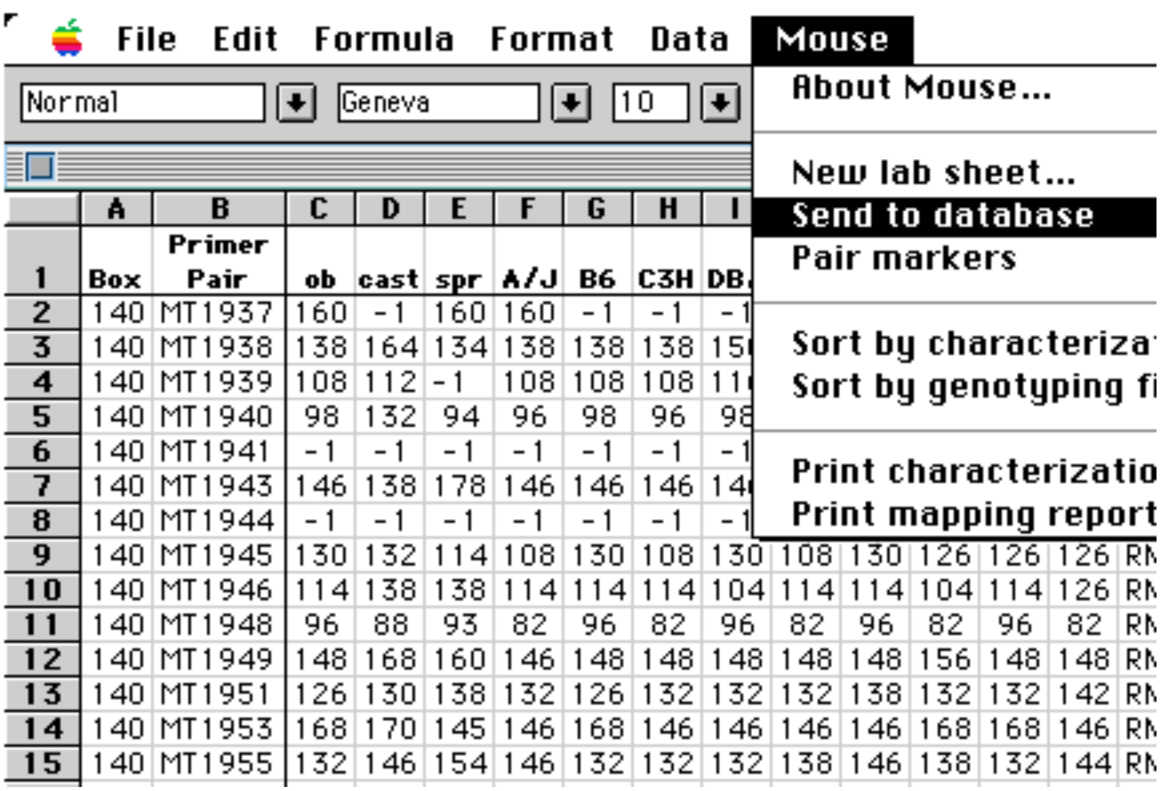

Figure 3: Portion of spreadsheet used in data entry for mouse geneti c mapping protocol

between the computer and the scientist than is offered by either behind-the-scenes processing or one way data transfer. For these situations we make extensive use of Microsoft Excel for data entry, viewing and manipulation. The spreadsheet has become a familiar and intuitive user interface, and is in fact the way that most of the scientists in the laboratory are accustomed to storing and organizing laboratory results.

We have created a set of external code modules ("plug-ins") that give Excel network access to MapBase and other UNIX tools. With the appropriate macros, these plug-ins allow custom spreadsheets to write, retrieve and update MapBase records. In this way a scientist can open up a spreadsheet that lists a subset of the sequences and the experimental results associated with them. Although the data appears to be local to the spreadsheet and can be copied, pasted, summarized, printed and otherwise manipulated in the usual spreadsheet manner, it is actually tied to the underlying data structures in MapBase. To add or edit data, the scientist makes the appropriate changes in the spreadsheet and chooses the "Send to database" menu command, which updates MapBase. A screenshot of one of our spreadsheets in action is shown in Figure 3 (previous page). This spreadsheet is used to enter the lengths of simple sequence repeats in various inbred strains.

In addition to the advantages of familiarity and ease of use, this approach has allowed us to incorporate data analysis and data integrity checking tools to the Excel spreadsheets in a simple and extensible manner. The most frequently-used Excel plug-in simply sends the contents of the spreadsheet as tab-delimited text over a TCP/IP socket to a waiting perl script. The perl script figures out what's to be done with the spreadsheet data, invokes the appropriate data analysis and database tools, adds or modifies the text and sends it back over the socket to the Excel plug-in which obligingly pastes it back into the spreadsheet. Adding new behavior to the spreadsheet is often as simple as updating the perl script.

A good example of this extensibility is our experience when we decided to add error checking to the genetic linkage mapping data spreadsheet. A frequent source of laboratory error occurs when laboratory technicians make typographical errors when they type the simple sequence repeat length inheritance patterns into the spreadsheet. The error is caught that night when a full MAPMAKER run is performed and the repeat is found to be unmappable, but by this time the data has already been entered into MapBase and the technician has put the experimental results away. We wished to catch errors at the point of data entry, while the experimental results were still fresh. While it was impractical to invoke a full MAPMAKER for each new simple sequence repeat entered, it was possible to write a quick and dirty program that roughly maps new repeats and catches most errors. By having the perl spreadsheet listener invoke this error-checking program, we were able to give technicians rough mapping feedback almost immediately and to catch the typographical errors before the data was entered into MapBase. Best of all, we did this without writing any new Macintosh code.

2.1.4 For database queries, use an E-mail interface.

E-mail is a particularly effective paradigm for posing database queries. Complex MapBase queries can take 20 to 30 seconds to complete. While this is not a particularly long wait, it is too long for an interactive session mediated by a graphical user interface, where an immediate response is expected. In contrast, scientists are accustomed to using E-mail to query each other, and they expect a delay of minutes to hours between sending out a request for information and receiving a response. Scientists can address *ad hoc* queries to MapBase via familiar E-mail software, Microsoft Mail, using a series of graphical forms we've designed. By filling out checkboxes and text fields, users of the system can set the conditions to satisfy and select the data fields to retrieve. They then send the form to the database and in less than a minute their query is answered by return mail. Textual data, such as status reports, is returned as word processor documents, while graphical information, such as maps, is returned in the form of Macintosh picture files. Tabular data, as one would expect, is returned as spreadsheet files. This E-mail system is also used for posing queries to MapBase over the Internet using a set of text-only forms. Figure 4 (next page) shows a portion of one of our E-mail forms. This one is used within the laboratory to obtain information about the progress of the genetic mapping protocol.

The E-mail query system is implemented using familiar UNIX tools. An alias called "genome_database" pipes incoming mail to a perl script that determines which query form is being used and where the query is coming from. Using this information, the perl script reformats the query form into a series of keyword/value pairs. This data is then passed to another perl script that queries MapBase and reformats the results in human-readable form. In some cases, when the user wishes to receive data as a spreadsheet or a picture file, the result text goes through an addition step in which it is piped through UNIX tools that reformat it as appropriate (these tools are written in C or perl). Finally the data is E-mailed back to the user.

The translation of queries from the graphical format used by Microsoft Mail to the text-format expected by the UNIX E-mail query system is accomplished by Microsoft Mail itself. Graphical forms sent outside the Microsoft Mail system are converted into a textual representation using rules that can be specified when the forms are designed. We specify conversion rules that are easily perl-parseable.

2.1.5 When all else fails, do it yourself.

We did of course encounter a small number of situations in which existing Macintosh software did not handle the job. The two examples that we encountered both involved entry of laboratory image data. In one case the solution was to capture the image via a video camera and interpret it using custom image analysis software. In another case the solution was to use a digitizing tablet to capture the positions of data points using a small Macintosh program and then to automatically paste the data into an Excel spreadsheet. Both these tasks were made easier by the use of Apple Events, which allow Macintosh programs to exchange data and coordinate their activities. In the case of the former task, the software that controls the video camera resides on a Macintosh while the software that interprets the image data resides on a DEC alpha/OSF 1. To integrate the two, we wrote a Macintosh daemon process that listens on a TCP/IP port for incoming messages from the image analysis program and forwards them, in the form of Apple Events, to the camera control program.

3. Conclusions

We have found the UNIX toolkit approach to be very helpful in designing and maintaining our informatics system. As the laboratory protocol has changed,

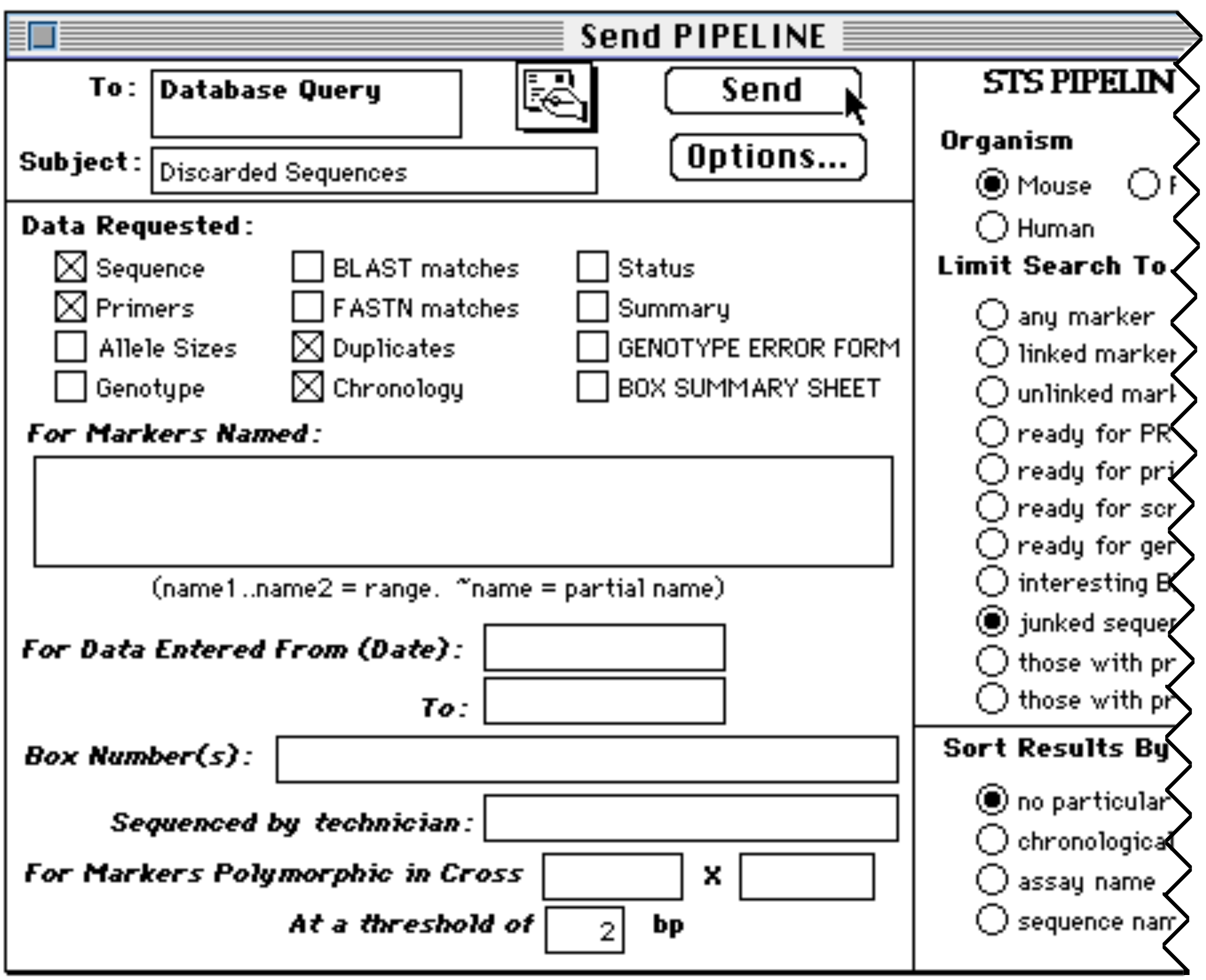

Figure 4: Portion of an E-mail form used for querying MapBase

we've been able to modify our system by adding or altering software modules or changing the order in which they execute. We have adopted the same approach to the design of our user interfaces. Instead of building our own interfaces from scratch, we have adapted existing software with which our users are already comfortable. The result has been a system that has allowed data throughput to grow by a factor of six (from 50 genetically mapped sequences per month to over 300/month) over a period of a year, and that enjoys a high level of user satisfaction.

Our approach is different from those taken by several other genome mapping groups. One approach, exemplified by the ACEDB database of the *C. elegans* genome is the "laboratory notebook" approach [Sulston *et al.,* 1992], in which the entire user interface is incorporated into one custom piece of software. An advantage of this system is that the interface is carefully crafted to fit the application. A disadvantage is that it is difficult to adapt the interface to meet changing laboratory needs. A more tool-oriented approach has been taken by the Chromosome 11 project, in which a relational database is used to integrate and control the activities of a number of data analysis and manipulation tools [Clark *et al.* 1994]. However in this case the decision was made to use a Apple Macintosh-based relational DBMS in order to take advantage of that system's graphical user interface, and this design decision has restricted the possibilities for automating information flow since the Macintosh OS does not provide the level of inter-process communication offered by UNIX. An approach similar to ours, but for a system to support a large-scale expressed sequence project, has been described by Kerlavage [Kerlavage *et al.* 1993]. They also build a pipeline of UNIX-based data manipulation tools drawing on a centralized database and interacting with scientists through Macintosh front ends. The major difference between their approach and ours is that their Macintosh interfaces are built on a single environment, Hypercard (Apple Computer), whereas we use different applications to present differing user interface paradigms.

Although our system was designed to meet the needs of a particular genome laboratory, our approach may be applicable in other situations in which it is necessary to integrate UNIX and PC environments.

4. References

Clark, SP, Evans GA, and Garner HR (1994). Informatics and Automation Used in Physical Mapping of the Genome. In "Biocomputing: Informatics and Genome Projects", Douglas Smith Ed., Academic Press, NY, pp. 13-49.

Dietrich W, Katz H, Lincoln SE, Shin H-S, Friedman J, Dracopoli NC, and Lander ES (1993). A Genetic Map of the Mouse Suitable for Typing Intraspecific Crosses. In "Genetic Maps, Locus Maps of Complex Genomes, Nonhuman Vertebrates", Vol 4, Stephen J. O'Brien, ed. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY. pp. 110-142.

Green ED, Olson MV (1990). Systematic screening of yeast artificial-chromosome libraries by use of the polymerase chain reaction. Proc Natl Acad Sci US. 87:1213-1217

Goodman N, Rozen S, and Stein L (1993). Requirements for a Deductive Query Language in the MapBase Genome-Mapping Database. Workshop on Programming with Logic Databases, Vancouver, BC, Oct. 30, 1993.

Goodman N (1994). An Object Oriented DBMS War Story: Developing a Genome Mapping Database in C++. In "Modern Database Management: Object-Oriented and Multidatabase Technologies", Won Kim, Ed., ACM Press, NY.

Lander ES, Green P, Abrahamson J, Barlow A, Daly M., Lincoln S, Newburg L (1987). MAPMAKER: an interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. Genomics 1: 174-181.

National Research Council, Committee on Mapping and Sequencing the Human Genome (1988) "Mapping and Sequencing the Human Genome", National Academy Press, Washington DC.

Kerlavage AR, Adams MD, Kelley JC, Dubnick M, Powell J, Shanmugam P, Venter JC, and Fields C (1993). Analysis and Management of Data from High_Throughput Sequence Tag Projects. in "Proceedings of the 26th Annual Hawaii International Conference on System Sciences", Trevor N. Mudge, Veljko Milutinovic and Lawrence Hunter, Eds. IEEE Computer Society Press. pp. 585-594.

Kernighan BW, and Plauger PJ (1976) "Software Tools", Addison-Wesley, NY, 1976.

Sulston J, Du Z, Thomas K, Wilson R, Hillier L, Staden R, Halloran N, Green P, Thierry-Mieg J, Qui L, Dear S, Coulson A, Craxton M, Durbin R, Berks M, Metzstein M, Hawkins T R, Ainscough R, and Waterston R (1992). The *C. elegans* genome sequencing project: A beginning. Nature 356:37-41.

Wall, L, and Schwartz RL (1991). "Programming perl", O'Reilly & Associates, Inc., Sebastopol, CA.

5. Biographies

Lincoln Stein is a MD/PhD pathologist and molecular biologist from Harvard with a persistent interest in software development including hypermedia-based medical education software, image analysis software, and desktop publishing tools. He is assistant group leader of the MIT Genome Center informatics core and takes the credit for much of the user interface design (or lack thereof) presented in this paper. **Andre Marquis** is a software engineer with a BA in Cognitive Science from the University of Rochester and is the implementor of the spreadsheet interface described in this paper. **Ert Dredge** is a software engineer with a BA in Biology from MIT; his contribution includes the automated error checking tools. **Mary Pat Reeve** is a software engineer with a BA in Biology from MIT, who is responsible for implementing much of the automated sequence analysis described in this paper. **Mark Daly** is a software engineer with a BA in Physics from MIT; his contribution includes the integration of MAPMAKER with MapBase. **Steve Rozen** is a PhD in Computer Science from New York University whose background includes the development of banking information systems on Wall Street. **Nathan Goodman** is a former professor of Computer Science at Harvard and former Chief Computer Scientist of Kendall Square Research Corp. He is the group leader of the MIT Genome Center informatics core and is responsible for the design and implementation of MapBase.