Human Computer Interface Analysis of Tier1/2/3 Bioinformatics Tools

Annette Greiner, Kristen Taylor

DOE JOINT GENOME INSTITUTE
US DEPARTMENT OF ENERGY
OFFICE OF SCIENCE

BEFORE YOU CODE Do A Five Step Makeover

Improve Usability in ONE DAY or less

Bioinformatics software tools perform sophisticated jobs. Creating interfaces that are powerful and yet easy to use and learn is not simple. Often, development cycles are short and user feedback is limited. Running a simple usability test on a prototype before any real code is written can ultimately save time. The results from a usability test can improve the ease of use and learnability of both web applications and programmatic interfaces. The following outlines a five-step process that can be completed in approximately one day.

STEP 1 - Create Usability Test Tasks

2 Hours

- **Interview Users** about the tasks they need to accomplish (four to six users is ideal). Do a rigorous needs assessment: observe how users execute their tasks currently; look for points where the user leaves the system to get or set data. Does the user use other applications to gather data to execute the task? Does the user look information up in a book?
- **Create a high-level list of the tasks** users would like to accomplish with the system. An example task might be "Compare the nucleotide sequences of two homologous genes." Avoid creating detailed outlines of steps (such as instructions about where to click). You want to observe how users get from start to finish, not tell them how to do it.

STEP 2 – Create "Low Fidelity" Prototype

2 Hours

Users execute the usability test tasks created in step one on "low fidelity" prototypes. We discuss three types of prototypes that can be created to help observe user interaction. These prototypes are "low fidelity" because very little time is invested in creating them. It is easy to modify them or throw them away and start over.

-**Paper Prototype:** use paper and sticky notes to create a mock-up of the interface. Use acetate overlays to show state changes.

-**Prototyping Software – Denim** is an informal web prototyping tool that lets users sketch interfaces and add functioning buttons and links. Developed at the University of Washington- Prof. James Landay - http://dub.washington.edu/denim/.

-**Stub Prototypes** for programmatic interfaces: A stub is a thin-layer interface that contains empty function calls. The calls would have the list of parameters but perform no action except to return the appropriate response to the user.

** As you design the prototype, look for ways to minimize the number of exposed interface controls. Before bringing the prototype to the user, try the usability test tasks yourself.*

Step 3 – Test Prototype on Users Attempting to Do Usability Test Tasks

2 Hours

-Use at least 3 users, preferably a mix of advanced and novice users.

-Observe, don't coach users, as they perform the usability test tasks.

-Note usability problems and gather user comments.

Step 4 – Triage Usability Issues and Modify Interface

1 Hour

-Review the results from all users.

-Prioritize issues found.

-Update usability test tasks and prototype.

Step 5 – Repeat steps 2 – 4

At this point the process can be considered completed. However, another round of testing users with an updated prototype will help verify that changes made were successful. The more iterations you can make with this process the more usable your interface will be.

TIER 1

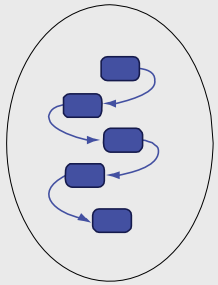
Application Example: BLAST Web Page

-This BLAST page has an interface that is simple to learn and use. Users perform one main task that follows a straightforward execution path.
- Parameter choices are clearly defined.
-The page has few options, which makes the density of interface controls low.

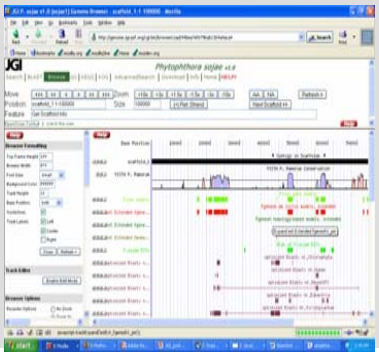
TIER 1:

Usability

- Attempting to increase flexibility by adding more options may complicate the process and learnability.
- Avoid complication by doing a rigorous needs assessment before creating usability test tasks.
- For example, this BLAST page encapsulates less frequently used options in an advanced settings button.



TIER 1: Interaction flow and usage patterns are simple.



TIER 2

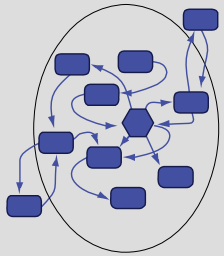
Application Example: Genome Browser

- A Genome Browser can be very difficult to learn and use. Users can perform multiple tasks and the execution path may even take users outside of the browser and then back.
- There are multiple layers of parameter choices and associated behaviors. Education is required to fully understand all the options available
- There are multiple levels of information. Density of interface controls is high.

TIER 2:

Usability

- Creating low-fidelity prototypes for this type of project can be overwhelming because of the multiple layers of information and high density of controls.
- Focus usability test tasks and prototype changes on areas that are critical or require more education before users can use them.
- For example, it is not obvious how the main area of the browser, at left, is used. Clicking on a gene model may expand the model or take the user to another application



TIER 2: Interaction flow and usage patterns are complex and may take users outside the application and back.

PIPELINE/WORKFLOW API

TIER 3:

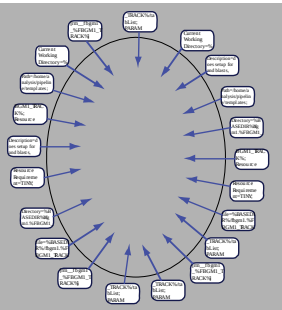
Application Example: Pipeline API

- It is important to consider usability when developing programmatic interfaces such as an Application Program Interface (API) for a pipeline.
- Pipeline interfaces require advanced users, and there is a learning curve associated with the API.
- The developer does not control the execution paths a user may choose, and an infinite number of execution path possibilities exist.

TIER 3:

Usability

- There is a tendency to over-engineer API calls. Adding features for special cases can decrease usability (and increase potential bugs.)
- Creating a stub as a prototype and observing how users use and interpret API names and parameters can help eliminate unnecessary features and redundancy and lead to a more usable API.



Tier 3: Data flow and usage patterns are varied and not controlled by the developer

LBNL-61198 Poster

UCRL

This work was performed under the auspices of the US Department of Energy's Office of Science, Biological and Environmental Research Program, and by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48, Lawrence Berkeley National Laboratory under contract No. DE-AC02-05CH11231 and Los Alamos National Laboratory under contract No. DE-AC02-06NA25396.