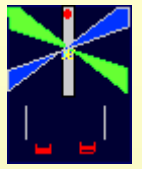
FacsXpert

July 8, 2004

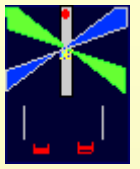
7th International Protégé Conference



- ***Builds protocols for studies with FACS instruments***

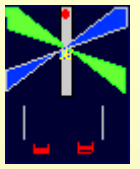
- ***Uses a modified Protégé-based architecture that promotes runtime extensibility for the end-user***

FacsXpert



- ***Built***
  - **At Stanford University's Herzenberg Lab**
  - **In close collaboration with Lee Herzenberg, James Tung, David Parks, Wayne Moore and other researchers in the Herzenberg Laboratory**
- ***With help from***
  - **Mark Musen, Ray Fergerson, Monica Crubezy, Natasha Noy and the Protégé community**

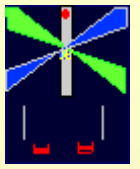
FacsXpert



- ***Presented by***
  - **Stephen Meehan, software engineer**
    Herzenberg Lab, Stanford University
    meehan@darwin.stanford.edu
  - **Since 1992, Stephen has pursued a "dream" of a "software democracy" architecture where:**
    - **An application derives the majority of its behavior from runtime interpretation of its UML design model**
    - **A user can extend this model while the application runs**

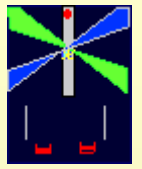
      Thus the model is more than a design artifact, it is runtime government by the people … for the people

Scope of Presentation



1. *The FACS domain and need for runtime extensibility*
2. *FacsXpert's approach to data structure extensibility*
3. *FacsXpert's approach to data integrity extensibility*
4. *FacsXpert's approach to GUI extensibility*

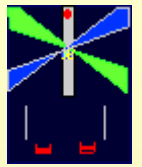
FACS instruments



Count and characterize fluorescent-labeled cells in suspension to

- **Monitor the progress of HIV infections by counting the number of CD4 T lymphocytes in blood from HIV-infected people**
- **Classify and stage tumors and to monitor bone marrow transplant survival**
- **Do the basic science and studies that underlie all these clinical advances**

FACS protocols specify what will be done in a FACS assay

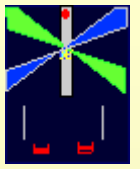


For each tube in the assay

- **Add up to 20 reagents, each linked to one of 12 fluorescent dyes**
- **Add one cell sample of up to 5 million cells**
- **Incubate and analyze with a multi-laser (Hi-D) FACS instrument**
- **Store the resulting data file (often about 5 megabytes)**

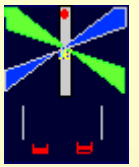
One assay often has 60 or more tubes

Sophisticated software is needed



- **Must compute a feasible combination of staining reagents (which "label" cells) by taking into consideration:**
  - ***Targeted species***
  - ***Targeted cellular markers***
  - ***Highly variable optical characteristics of fluorescent elements***
  - ***Optical detection capabilities of a FACS instrument's configuration***
  - ***Fluorescence interference between fluorescent elements***
  - ***Purpose of cell labeling: to gate, dump or discover cell populations***
  - ***Availability of inventory***
  - ***Bio-chemical affinities between staining steps***

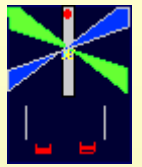
Sophisticated software is needed



- **Pipetting guide's**
  - ***Reagent and specimen amounts must follow highly variable dilution recommendations***
  - ***Organization must cater to pipetting technician work flow which differs from the planning work flow of the scientist***
- **Final protocol knowledge must interoperate with**
  - ***Instrument software (offline as well as real time)***
  - ***Analysis software***
- **GUI for decision making must be highly intuitive**

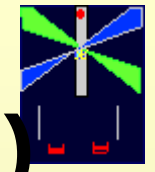
  Senior FACS scientists are often somewhere between computer naïve and totally computer-phobic

Highly extensible software is needed



- FACS vocabulary and “best practices” are
  - Minimally standardized
  - Rapidly changing
- From day to day, FACS scientists cannot predict
  - What questions they will be asking
  - What materials they will be working with
- Hence, their “research planning” software applications must be highly flexible and customizable

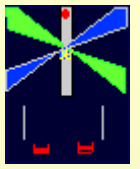
Highly extensible software (cont.)



- **THUS … scientists (in addition to knowledge/software engineers) must be able to extend**
  - **Data structures (classes, attributes, etc)**
  - **Data integrity (rules that check knowledge inputs)**
  - **The GUI**

  … without crashing the current application or future upgrades!!!
- ***Protégé provides the foundation for such a runtime evolvable system***
- ***Commercial alternatives (e.g. Oracle, Rational Rose) suffer from higher financial cost, higher sys admin cost, closed source and design-time centricity***

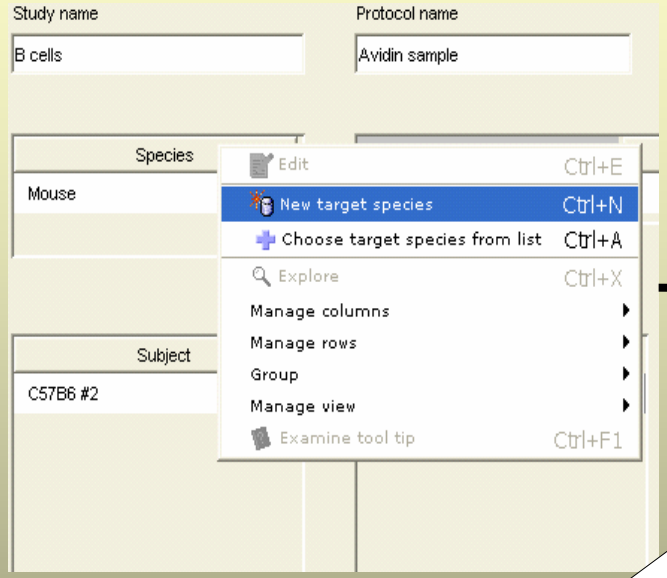
Data structure extensibility



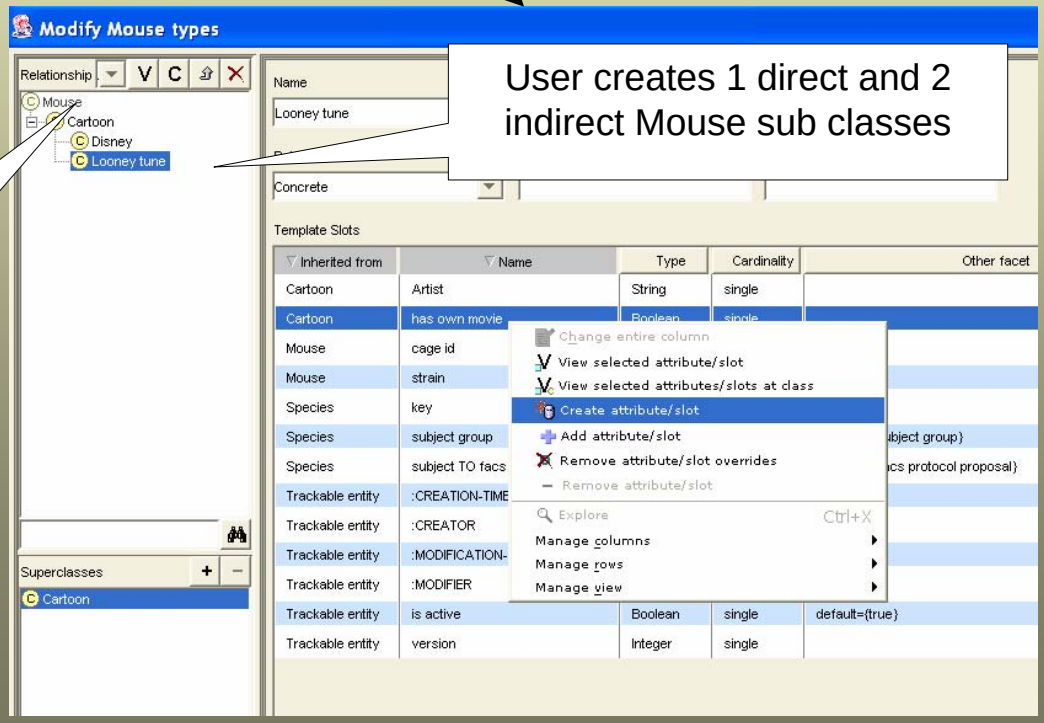
- **Protégé supports data structure extensibility through object-oriented sub classing**
- **However, the its model editor GUI overwhelms the scientist by exposing the whole model**

FacsXpert required a model editor that only focuses on a specific part of the model *at the specific time* that it is relevant to the application

FacsXpert solution for data structure extensibility



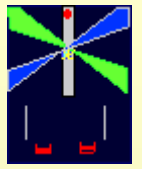
A class-type slot widget that supports a create action which limits the model editor's focus to the slot's allowed parents



User creates 1 direct and 2 indirect Mouse sub classes

Pop up focuses on species class hierarchy starting at Mouse

FacsXpert solution for data structure extensibility



- **A small # of programming idioms ready *FacsXpert* for such extensions; for example:**
  - ***Creating instances***

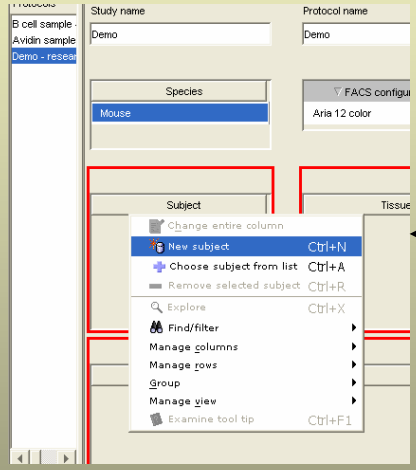
```
myProject.createInstance( null,
    DisplayUtilities.pickCls( null,
       Collections.singletonList( designTimeLeafClass
  )))
```

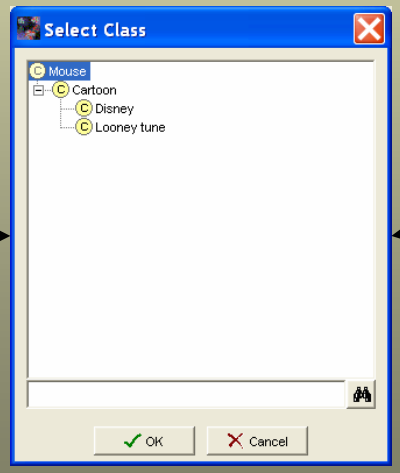
  - ***Querying class type***

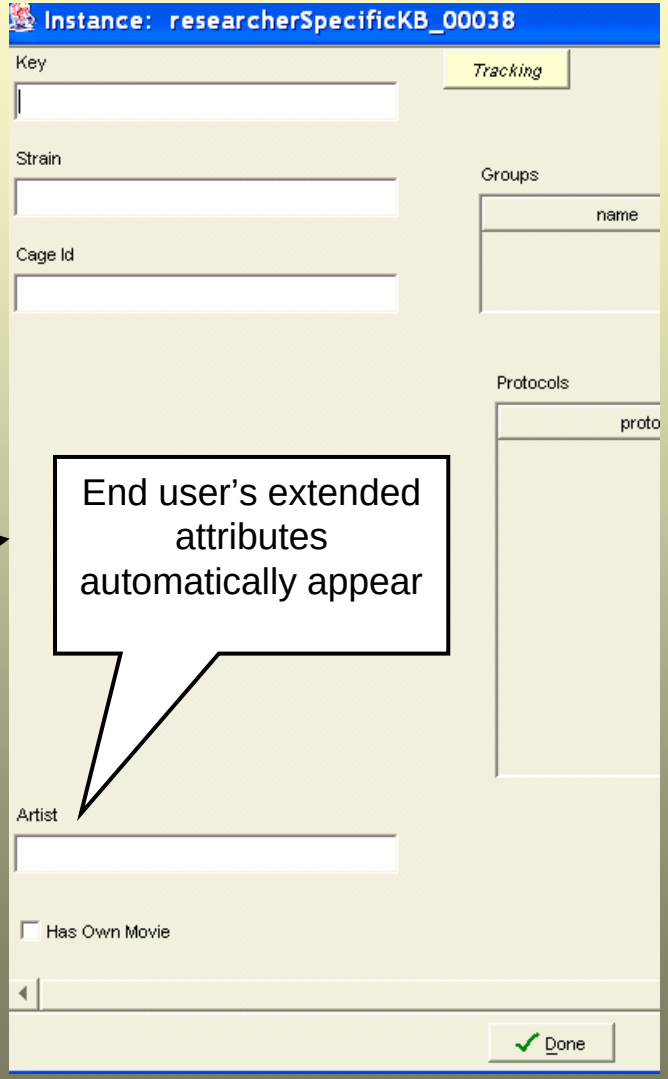
```
designTimeLeafClass .equals( unknownClass ) ||
  unknownClass.hasSuperclass( designTimeLeafClass )
```

Instantiating a new subject

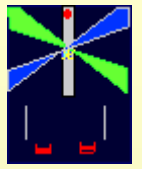
The programming idiom detects the need to query for sub classes to the mouse class







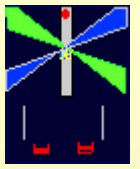
Darwin “crash-proofs” data structure extensibility



We built a tool named Darwin that guards model evolution by handling model editions which break hard-coded expectations

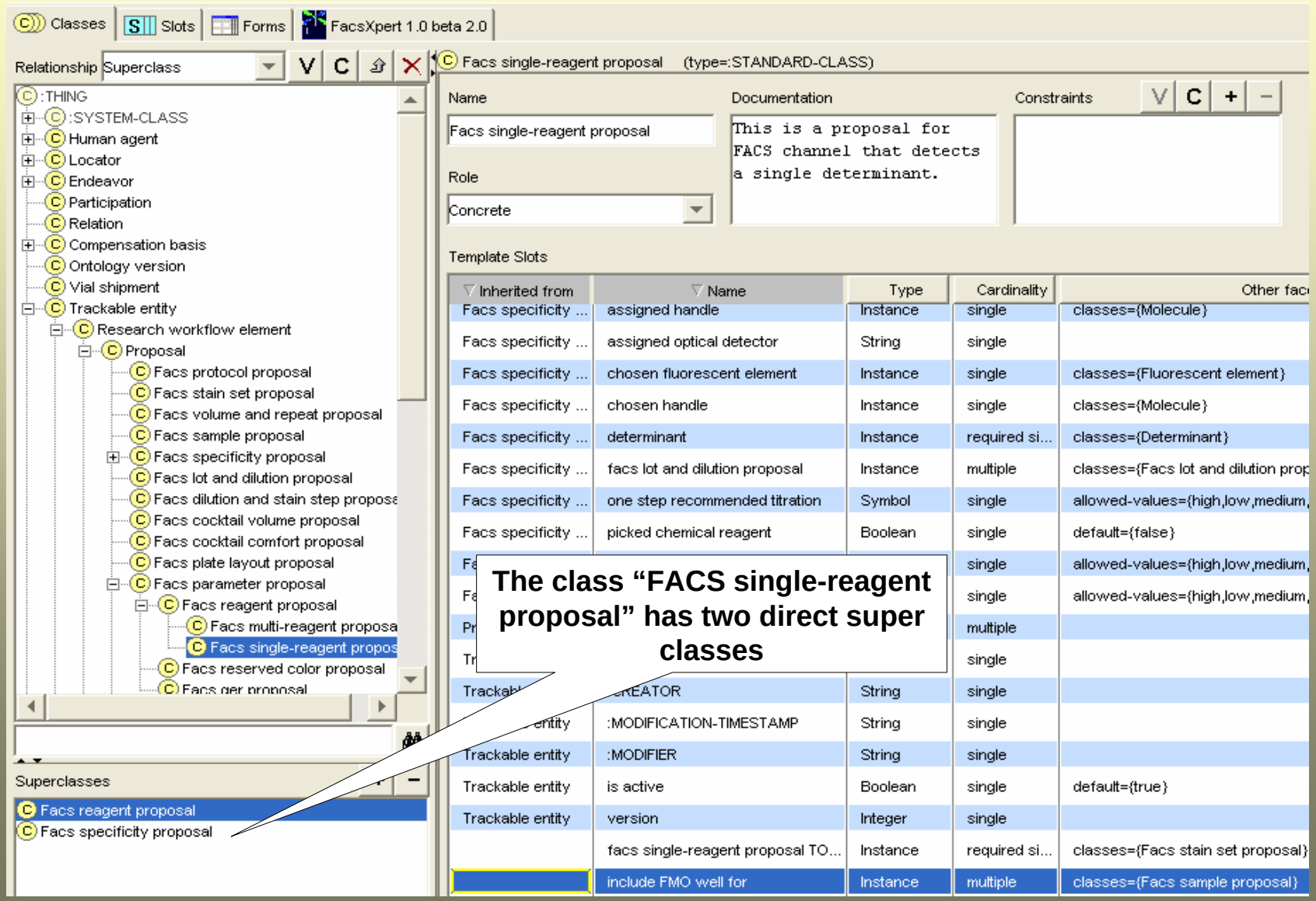
- **For any given class, Darwin prevents the “extinction” of one or more slot associations and one or more slot facets**
- **One can set 1 of 2 “watch dogs” to monitor changes to “endangered” model elements:**
  - ***Golden retriever*** **barks when user changes such an element and allows rollback (for engineers)**
  - ***Doberman Pincher*** **prevents user from making the change (for end-users)**

Darwin's JAVA generator

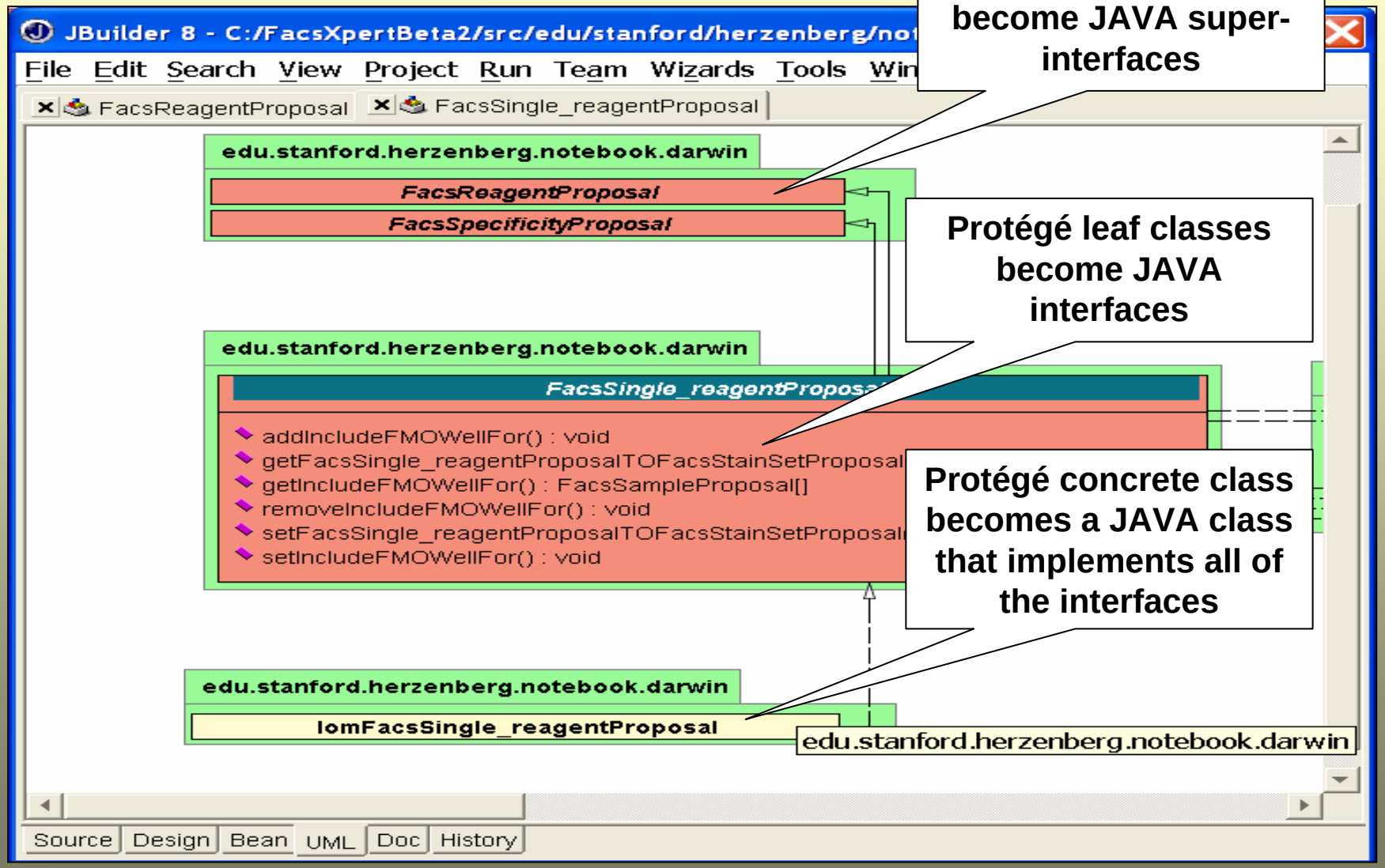


- **Darwin generates JAVA modules that benefit client code by**
  - ***Establishing a simple and consistent idiom for type-safe access to the endangered portions of the Protégé model***
  - ***Adding JAVA compiler checking as a means of tracking model dependencies***
- **Darwin translates**
  - **All Protégé classes into JAVA interfaces hence supporting multiple inheritance**
  - **Concrete Protégé classes into JAVA classes that implement all associated JAVA interfaces**

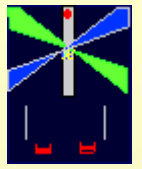
From Protégé class…



…To JAVA


JBuilder 8 - C:/FacsXpertBeta2/src/edu/stanford/herzenberg/not
File Edit Search View Project Run Team Wizards Tools Win
FacsReagentProposal
FacsSingle_reagentProposal
edu.stanford.herzenberg.notebook.darwin
FacsReagentProposal
FacsSpecificityProposal
edu.stanford.herzenberg.notebook.darwin
FacsSingle_reagentPropos
addIncludeFMOWellFor() : void
getFacsSingle_reagentProposalTOFacsStainSetProposal
getIncludeFMOWellFor() : FacsSampleProposal[]
removeIncludeFMOWellFor() : void
setFacsSingle_reagentProposalTOFacsStainSetProposal
setIncludeFMOWellFor() : void
edu.stanford.herzenberg.notebook.darwin
lomFacsSingle_reagentProposal
edu.stanford.herzenberg.notebook.darwin
Source Design Bean UML Doc History
Protégé super classes become JAVA super-interfaces
Protégé leaf classes become JAVA interfaces
Protégé concrete class becomes a JAVA class that implements all of the interfaces

Data integrity extensibility



Primary goal

- **Runtime extensibility of constraints that check knowledge inputs (hereafter *checks*). *Known to the Protégé community as "knowledge acquisition"***

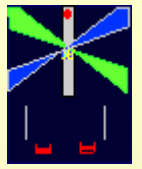
Out of scope

- **Runtime extensibility of constraints that**
  - **infer new knowledge**
  - **validate ontologies (consistency checking, disjointedness, semantic imbalance etc.)**

In practical terms…

- **Allow scientists to author and authorize checks while FacsXpert runs**

Requirements



A. Decouple check inquirers from check providers

B. Support reuse for both of the above actors:

   A. Inquirer: "Is anything wrong?"

   B. Provider:

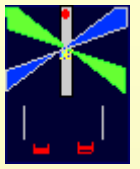
      A. Domain independent

      B. Parameter-izable check or check template; user completes parameters when "binding"

C. Make checks model associated *and* oriented

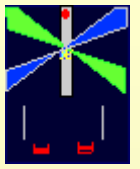
D. Integrate checks with Protégé forms & pick lists

Requirements



E. Make checks scaleable
   A. Focus on limited instance context
   B. Incremental execution
F. Stay in sync with model re-factoring
G. Support stepwise debugging
H. Represent constraints in model
   A. Start at :CONSTRAINT super class
   B. Support inquirer querying of characterizations such as: advisory, warning, error and fatal

Requirements



I. Allow user to add checks while FacsXpert runs

J. Support end user, ad hoc check authoring

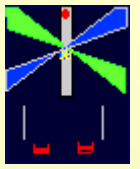
K. Allow computability of dependencies, conflicts & bottlenecks

   A. Prevent conflicts between checks

   B. Prevent conflicts between checks and model facets and onto-clean meta classes/slots

L. Have reasoning agents that use computability
   (otherwise prior requirement is “pie in the sky”)

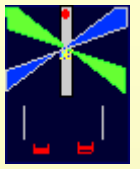
Data integrity extensibility



Engineering options (in 2002)

1. **Reuse an existing runtime computable grammar**
   - ✓ **Pal with EzPAL**
   - ✓ **JESS**
   - ✓ **OCL**
2. **Invent a new "non programmatic" grammar; implementation options included:**
   1. **Translate invented grammar to accepted grammar to reuse interpreter**
   2. **Interpret the invented grammar directly**
3. **New JAVA validity checking framework that promotes "parameterized check templates"**

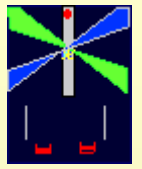
We took choice 3



... and met 7 of 12 requirements

- ✓ **A. Decouple check inquirers from check providers**
- ✓ **B. Support reuse in both of the above actors**
- ✓ **C. Make checks model associated *and* oriented**
  - o **By sub classing Darwin**
- ✓ **D. Integrate checks with Protégé forms & pick lists**
- ✓ **E. Make checks scaleable**
- ✓ **F. Stay in sync with model re-factoring:**
  - o **by sub classing Darwin class**
- ✓ **G. Support step-wise debugging**

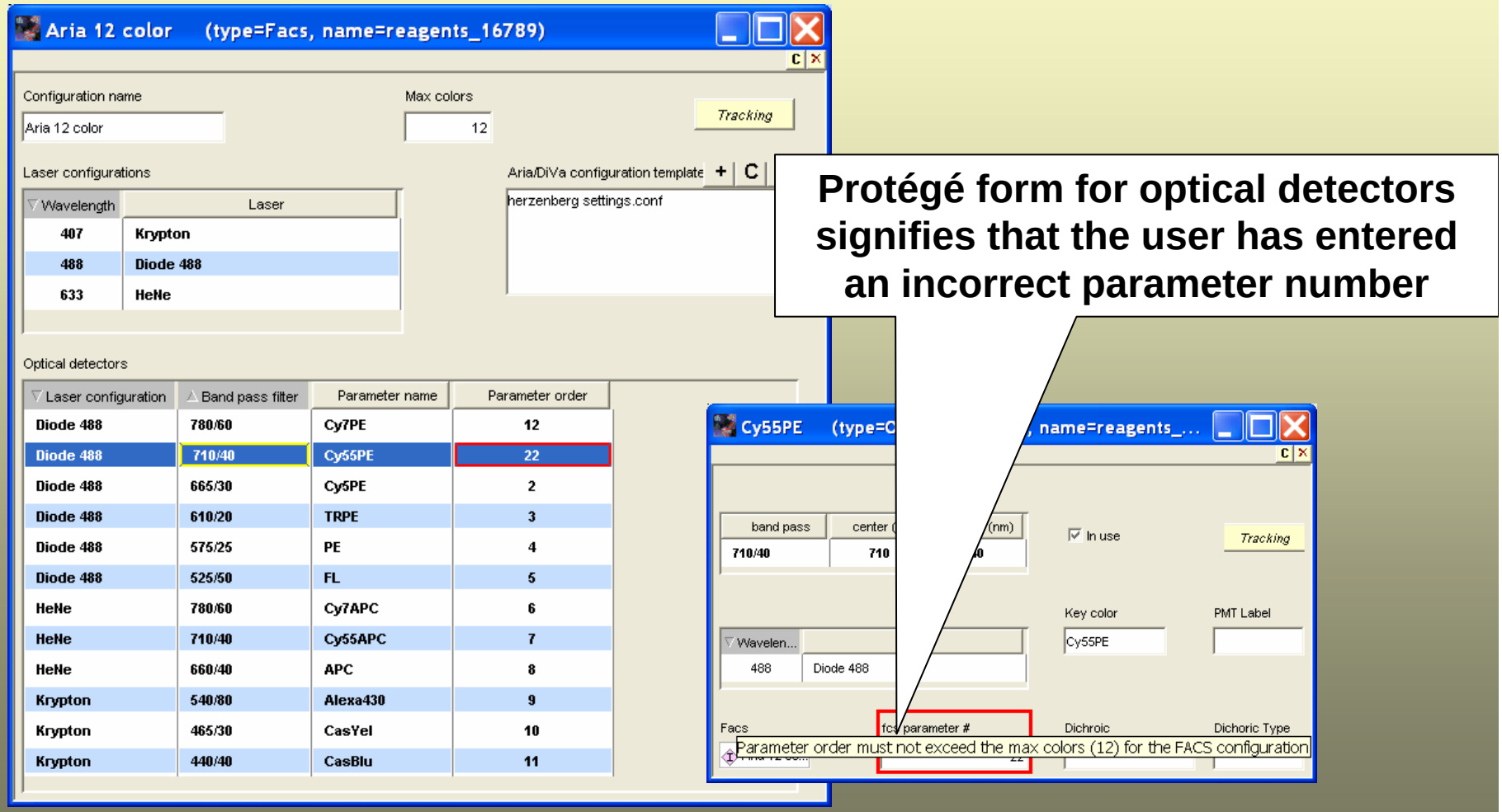
An example of validity checking



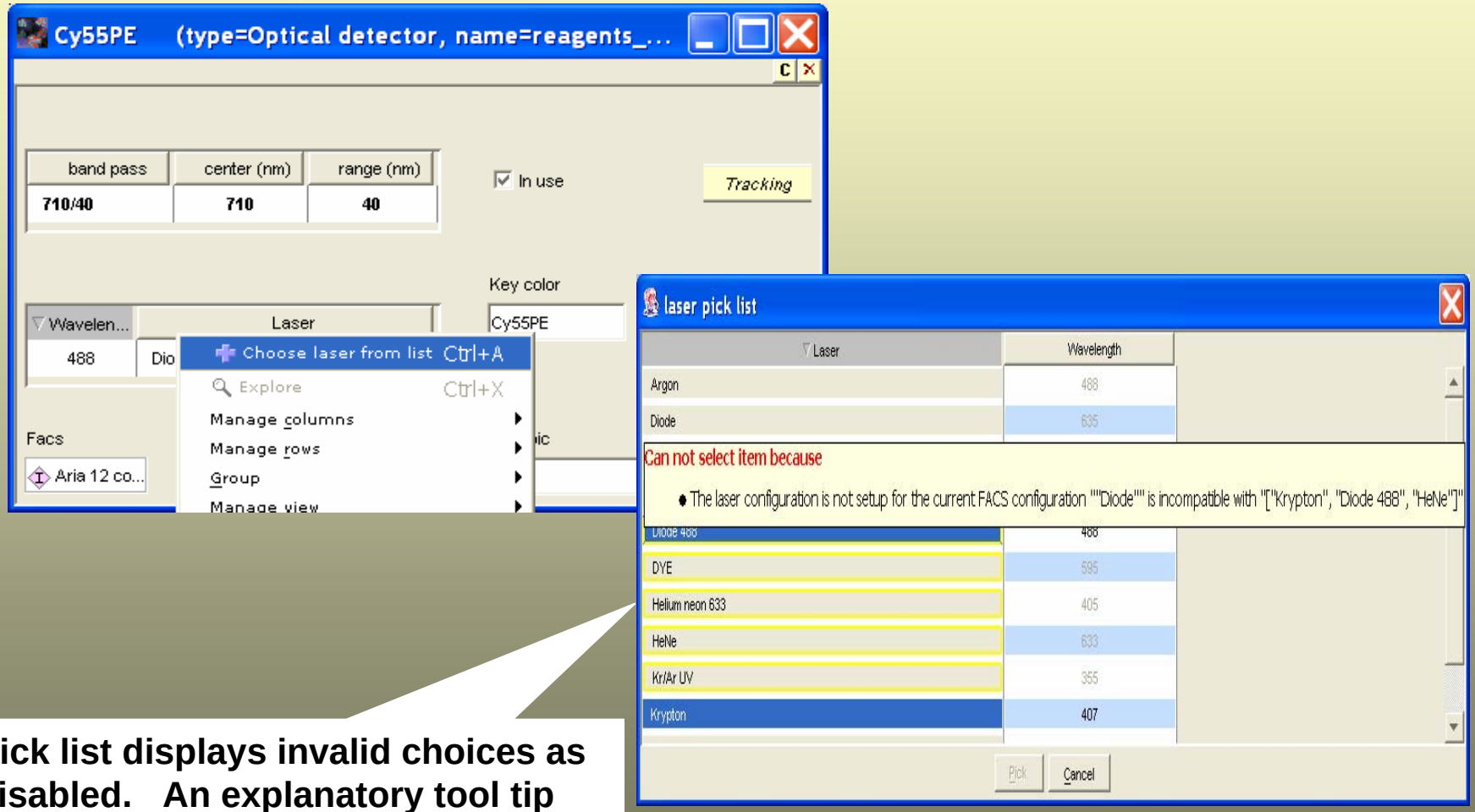
Optical detector class contains validity checks such as

- **Number assigned for a FACS parameter must be**
  - **Between 1 and the number of colors for the parent FACS configuration**
  - **Unique amongst sibling optical detectors**
- **The laser configuration associated with a detector must also be associated with the FACS configuration parent**

Checking integrity for parameter number value

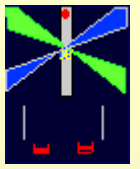


Checking integrity of Laser configuration relationship



Pick list displays invalid choices as disabled. An explanatory tool tip appears when the mouse is hovered over the disabled item.

What are our next steps?



Meet remaining requirements

H. Represent constraints in model

I. Allow user to add checks while FacsXpert runs

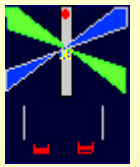
J. Support end user, ad hoc check authoring

K. Allow computability of dependencies, conflicts & bottlenecks

L. Have reasoning agents that use computability, otherwise prior requirement is pie in the sky

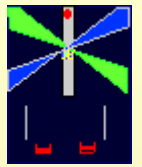
Current approach is limited:

- **H and I are easily achievable, but not J, K and L**
- **At best it will become hidden plumbing for approach that hits all 12 requirements**



And then there's the GUI

Protégé has GUI extensibility… but frankly

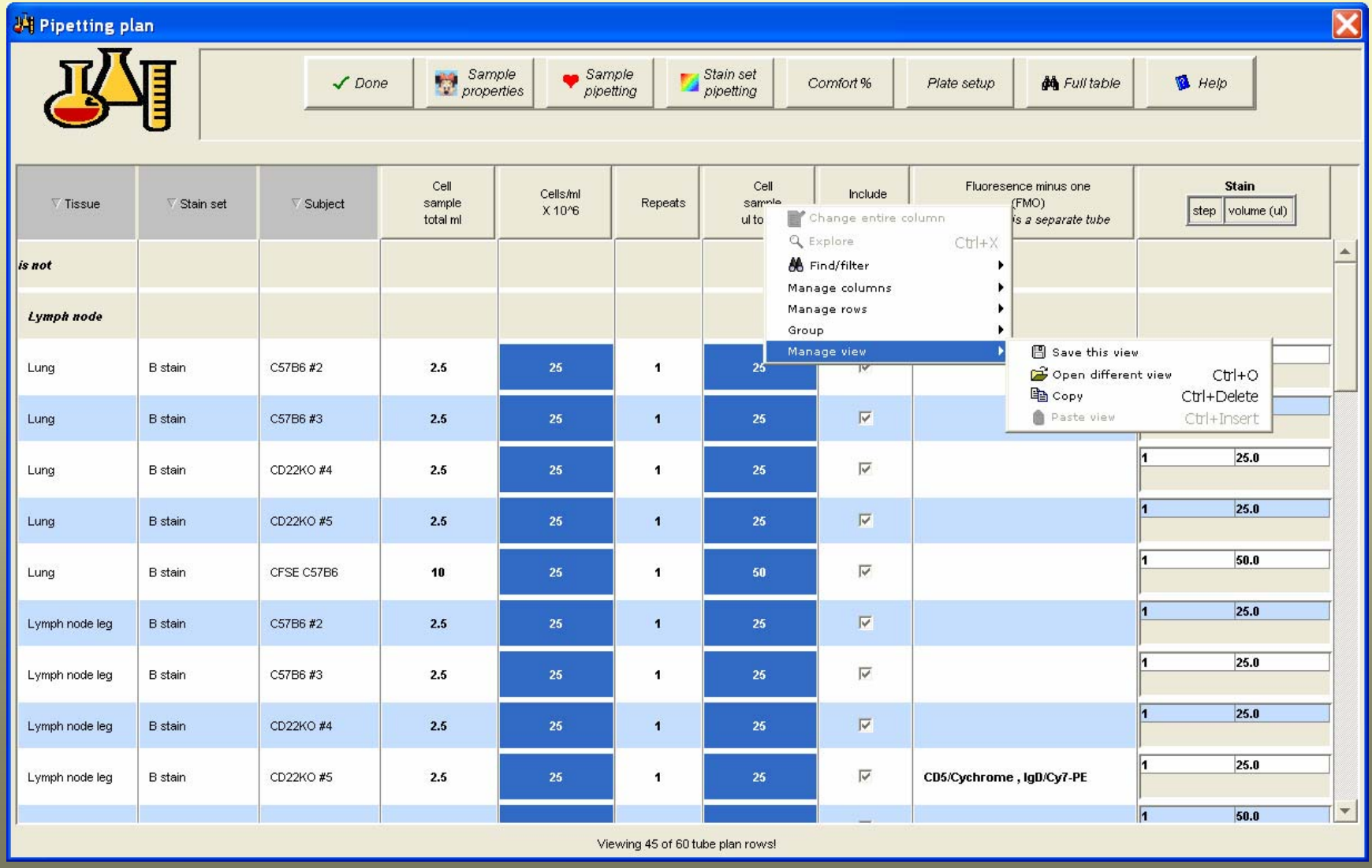


FACS scientists and the Protégé GUI were like oil and water

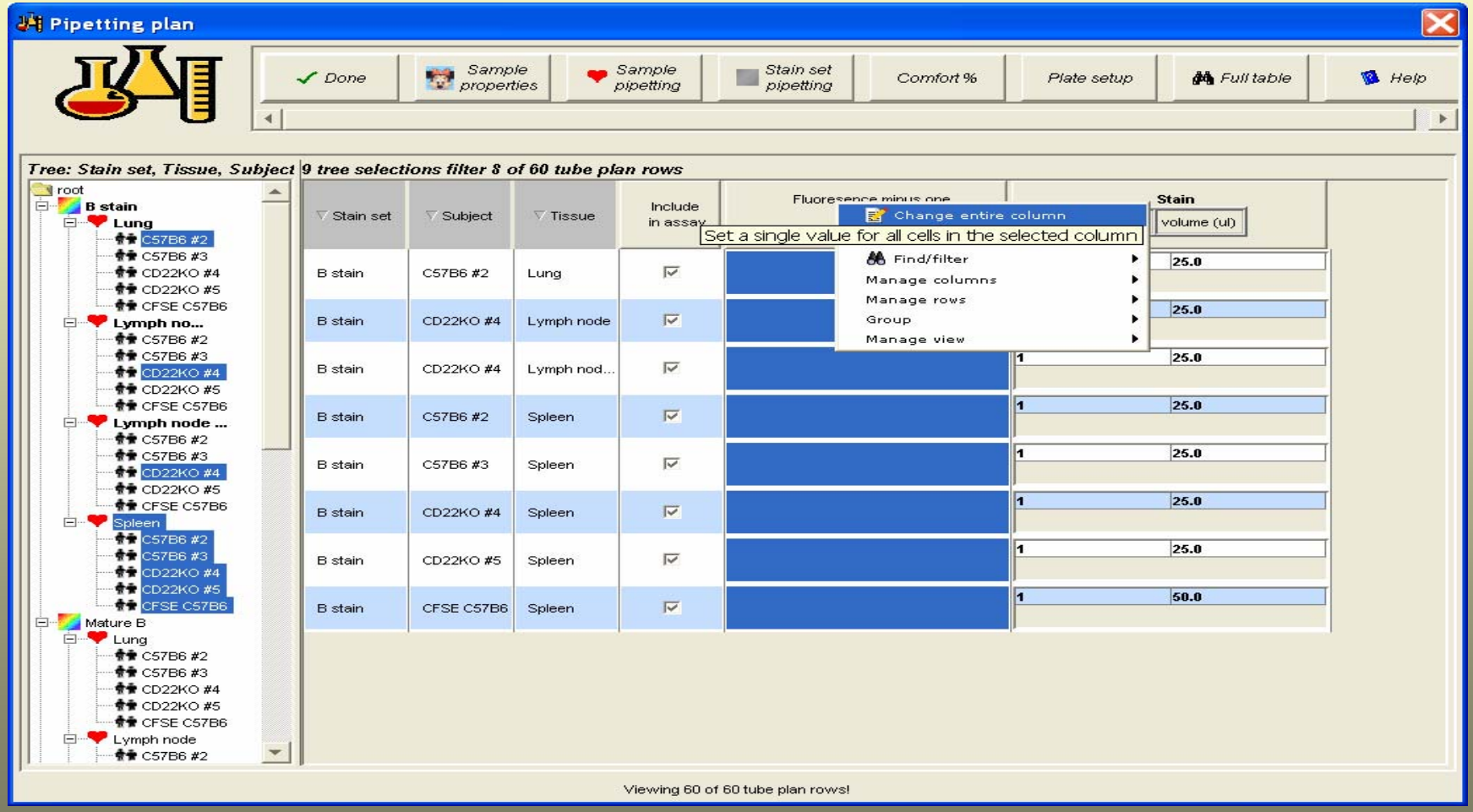
- They were confused by the slot buttons with +, -, C,V, X … and they balked at the dialogs
- They begged for a highly customizable Xcel-like widget (grid) for manipulating tables of data
- Then, after many iterations of developing this table widget, they ultimately decided that navigation and group-based modifications of items in the table had to be further simplified.

They wanted to access the table via a tree!

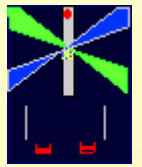
From a table (linear) view to…



…To a tree (hierarchical) view

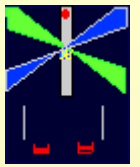


Xpert grid / table supports



- **User customizability**
  - Customize column: names, sizes, display order and sort order
  - Filter rows
- **Persistence of user customizations**
  - Automatically remember and use the most recent customizations
  - Allow the user to save customizations
  - Allow the user to save and retrieve sets of customizations into property files
- **Tree-based read / write access**
  - Define a tree structure to summarize elements of the underlying table
  - Select any combination of tree nodes
  - See the column entries sieved by the node selections
  - Modify all cells in a column for the sieved rows

GUI extensibility



- Added pick list plug-in framework
- Added system menu plug-in and overrides
- Added numerous other slot widgets