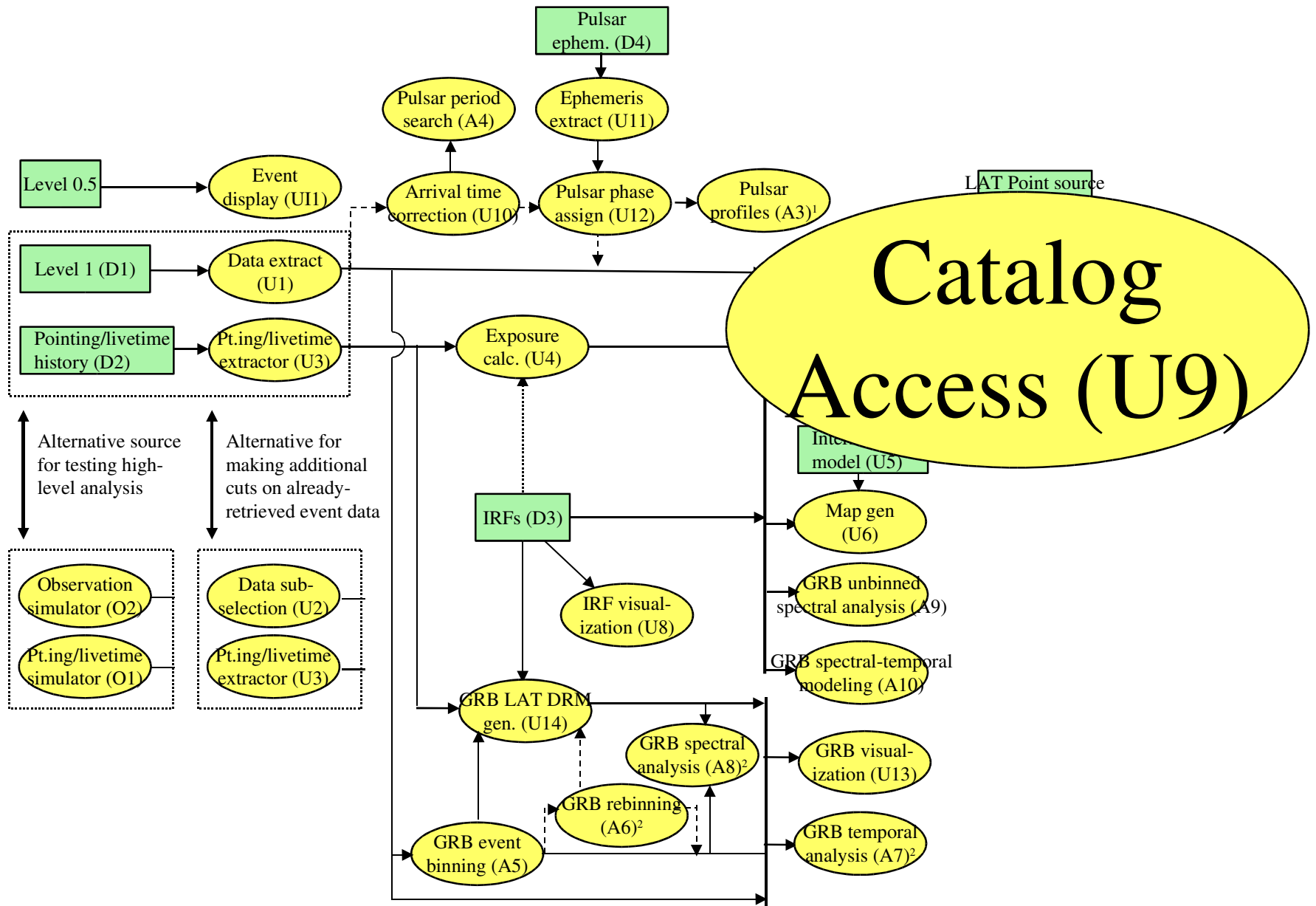
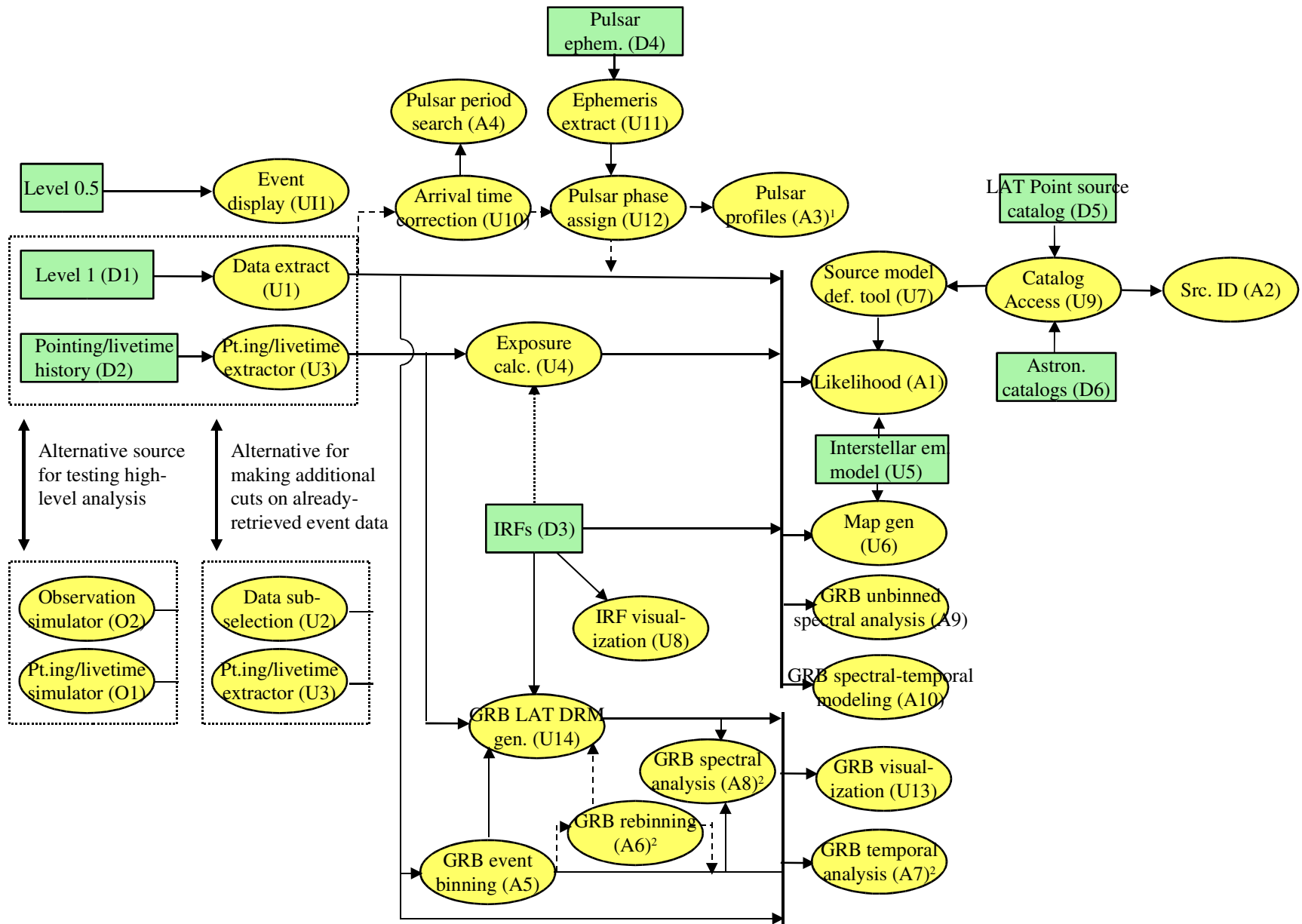


# The Standard Analysis Environment



# The Standard Analysis Environment



**Purpose** (from the design document [http://glast.gsfc.nasa.gov/ssc/dev/catalog\\_tools/u9def](http://glast.gsfc.nasa.gov/ssc/dev/catalog_tools/u9def))

The U9 functionality, as originally defined in the Standard Analysis Environment (SAE) design document (September 2002), is that of **providing access from SAE tools to (a) the host of astronomical catalogs available from websites such as CDS and NED and (b) catalogs of GLAST origin.**

Examples for catalogs of non-GLAST origin are the Third EGRET Catalog, catalogs of flat spectrum radio quasars, catalogs of millisecond pulsars, catalogs of OB associations etc. Examples for GLAST catalogs are first of all the GLAST pointsource catalog, but there may also be other sub-catalogs generated.

**U9 shall provide the methods such that tools like the model definition tool U7 or the source identification tool A2 can import a catalog into an internal representation with which they can work on their respective tasks.**

The catalog import method for a particular catalog will in general be dependent on the version of the catalog and therefore change with time.

In order to ensure that the appropriate catalog version be available, **the tool shall also provide methods to save and load a copy of the imported catalog in a FITS format** corresponding to the internal catalog representation mentioned above. This GLAST-internal representation of the catalog can then optionally be distributed with the tool.

## U9 Implementation

Two classes:

- 1) A class holding the description of an individual catalog column:
  - named “quantity”
  - three types: Numerical (i.e. double)
    - String
    - Vector (forms a group of other non-vector quantities)
  - in addition to type, name, and physical unit information, the class holds
    - a) the selection criteria applied to the individual quantity
    - b) a UCD and references to error quantities (if available)
  
- 2) A class holding the catalog description and contents
  - named “catalog”
  - the catalog description is a vector of “quantities” (class 1 above)
  - the catalog contents is stored in two `std::vectors`, one for numerical and one for string quantities
  - “catalog” provides access methods, selection methods, methods for importing from the original catalog files, methods for saving to and loading from fits files, methods for finding value ranges, sorting

## U9 Implementation

Access to the catalog information:

a) via the quantity name

b) via the UCD

c) via generic access methods (for quantities common to all catalogs such as RA and DEC)

d) via error access methods (access the error of a quantity via the name of the quantity)

f) via a vector quantity name

← Not in initial versions

Selection:

a) upper and lower bounds

b) accepting or rejecting specific values

c) elliptic regions for RA, DEC or l, b

d) free format selection formula with parser (lower development priority)

← Not in initial versions

## Status

Aymeric Sauvegeon has checked in Version 0 of U9 to the SLAC CVS repository this morning:

Large part of the functionality implemented.

Development of client applications can start (A2, U7).

Remaining functionality will be implemented well before DC2.

Need WCSTools to be integrated into astro package for precession feature!

## Main members of the quantity class

```

/*****
std::string m_name;          // The name of the quantity, e.g., "hr1"
std::string m_comment;     // e.g. " hardness ratio 1-3keV/3-6keV"
std::string m_ucd;
    // Unified contents descriptor (if available, "" otherwise)
    // follows either UCD1 standard or UCD1+ (still t.b.d.)
std::string m_format;     // format used in CDS database
QuantityType m_type;
std::string m_unit;
    // The unit description, "1" if dimensionless, "" if string
int m_index;
    // Where to find the quantity in the m_strings or m_numericals
    // catalog members array
bool m_isGeneric;
    // True if the quantity is part of the set of quantities
    // common to all catalogs
bool m_toBeLoaded;
    // True if data corresponding to this quantity must be loaded into memory
std::string m_statError;
    // For numerical quantities: the name of the quantity which contains
    // the statistical error of this quantity;
    // empty if unavailable or for strings
std::string m_sysError;
    // For numerical quantities: the name of the quantity which contains
    // the systematic error of this quantity;
    // empty if unavailable or for strings
std::vector<std::string> m_vectorQs;
    // If type == VECTOR, this vector of strings contains the ordered list
    // of quantity names which constitute the vector elements.
    // All vector elements have to be numerical quantities.
    // If type!=VECTOR, vectorQs is empty.

```

## Members of the quantity class (continued):

```

// selection criteria
//-----█

std::vector<std::string> m_excludedS;
    // for numerical: empty
    // for string   : stores the list of values leading to exclusion of a row
std::vector<std::string> m_necessaryS;
    // for numerical: empty
    // for string   : stores the list of values necessary for row inclusion
double m_lowerCut;
    // for string   : undefined; default == 1E-99
    // for numerical: stores the minimum value necessary for row inclusion
double m_upperCut;
    // for string   : undefined; default == 1E99
    // for numerical: stores the maximum value necessary for row inclusion
std::vector<double> m_excludedN;
    // for string   : undefined; default: empty
    // for numerical: stores the list of values leading to exclusion of a row
std::vector<double> m_necessaryN;
    // for string   : empty; default: empty
    // for numerical: stores the list of values necessary for row inclusion

double m_precision; // test for equality used for m_excludedN, m_necessaryN
bool   m_rejectNaN; // if true the NaN values are not selected

```

The heart of U9: the class “catalog”  
contains

- Methods giving general information
- Methods for importing, saving, loading
- Methods for accessing data
  - Catalog definition
  - catalog contents in memory (ignoring selection criteria)
  - possible values or value range for a given quantity (ignoring selection criteria)
  - catalog contents in memory with selection criteria applied
  - possible values or value range for a given quantity with criteria applied
- Methods for selecting data
- Methods for sorting

### Example 1: load the Veron catalog into memory, identify the 50 closest sources

```
int n = 50;
int numRows;
double zCut;
double minflux, maxflux;

catalog *myCatalog = new catalog();
numRows = myCatalog->importWeb("VeronVeron2003");
// the catalog names are predefined,
// the import method "knows" them
if(numRows <= 0) exit; // in this example, error handling is unforgiving
myCatalog->sortAscend("z");
myCatalog.getNValue("z", n-1, &zCut);
myCatalog.unSetCuts(); // make sure no other cuts are applied
myCatalog.setLowerCut("z", zCut); // this sets and applies the cut

minflux = myCatalog.minSelVal("F6cm");
maxflux = myCatalog.maxSelVal("F6cm");
cout << "max flux = " << maxflux << ", min flux = " << minflux << std::endl;
myCatalog.saveSelected("veron50brightest.fits");
```

**Example 2:** load a circular region around RA = 120.1, DEC=25.3 with 5 diameter from the Veron catalog into memory and identify the brightest source (at 11 cm)

```
std::string name;
int numRows;

catalog *myCatalog = new catalog();
myCatalog->importDescriptionWeb("VeronVeron2003");
myCatalog->setSelEllipse(120.1, 25.3, 2.5, 2.5, 0.);
numrows = importSelected();
if(numRows <= 0) exit; // in this example, error handling is unforgiving
myCatalog->sortDecend("F11cm");
getSelObjName(0, &name);
cout << "Brightest object is " << name
      << " at RA = " << selRA_deg(0)
      << "      DEC = " << selDEC_deg(0);
```

## Catalogs implemented in version 0:

```
"EGRET3 sources", "J/ApJS/123/79/3eg", // 271 rows (18 columns)
"EGRET3 fluxes", "J/ApJS/123/79/fluxes", // 5245 rows (10 columns)
"EGRET3 periods", "J/ApJS/123/79/table1", // 169 rows ( 6 columns)
"ROSAT 1RXS" , "IX/10A/1rxs"};
```

This list will continuously grow. By DC2 it will hopefully be quite complete.

Send requests for specific catalogs to [petry@milkyway.gsfc.nasa.gov](mailto:petry@milkyway.gsfc.nasa.gov)