StAF Architecture
Standard Analaysis Framework

Craig E. Tull
NERSC - LBNL

ATLAS Software Week

CERN

May 20, 1999

HENP Computing Challenges

Experiment	Data	Compute
E895 (AGS)	10 TB/yr	600 SPECint95
BaBar (SLAC)	400 TB/yr	5,000 SPECint95
STAR (RHIC)	266 TB/yr	10,100 SPECint95
PHENIX (RHIC)	700 TB/yr	8,500 SPECint95
D0 Run II (FNAL)	280 TB/yr	4,075 SPECint95
CDF Run II (FNAL)	464 TB/yr	3,650 SPECint95
ATLAS (LHC)	1100 TB/yr	2,000,000 SPECint95

Experiment	Contries	Institutes	Collaborators	Time Frame
E895 (AGS)	3	12	49	2000
BaBar (SLAC)	9	85	600	2010
STAR (RHIC)	7	3	400	2010
PHENIX (RHIC)	10	41	400	2010
D0 Run II (FNAL)	11	77	500	2005
CDF Run II (FNAL)	8	41	490	2005
ATLAS (LHC)	34	144	1700	2015

What is a Framework?

- **Gamma, et al., Design Patterns**

"When you use a toolkit, you write the main body of the application and call the code you want to reuse. When you use a framework, you reuse the main body and write the code it calls."

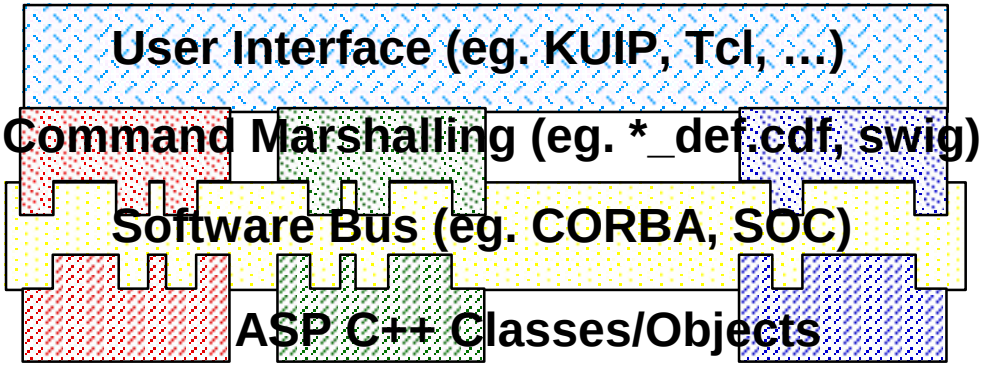
"Not only can you build applications faster as a result, but the applications have similar structures. They are easier to maintain, and they seem more consistent to their users. On the other hand, you lose some creative freedom, since many design decisions have been made for you."

"If applications are hard to design, and toolkits are harder, then frameworks are hardest of all. ... Any substantive change to the framework's design would reduce its benefits considerably, since the framework's main contribution to an application is the architecture it defines. Therefore it's imperative to design the framework to be as flexible and extensible as possible."

StAF Design Features

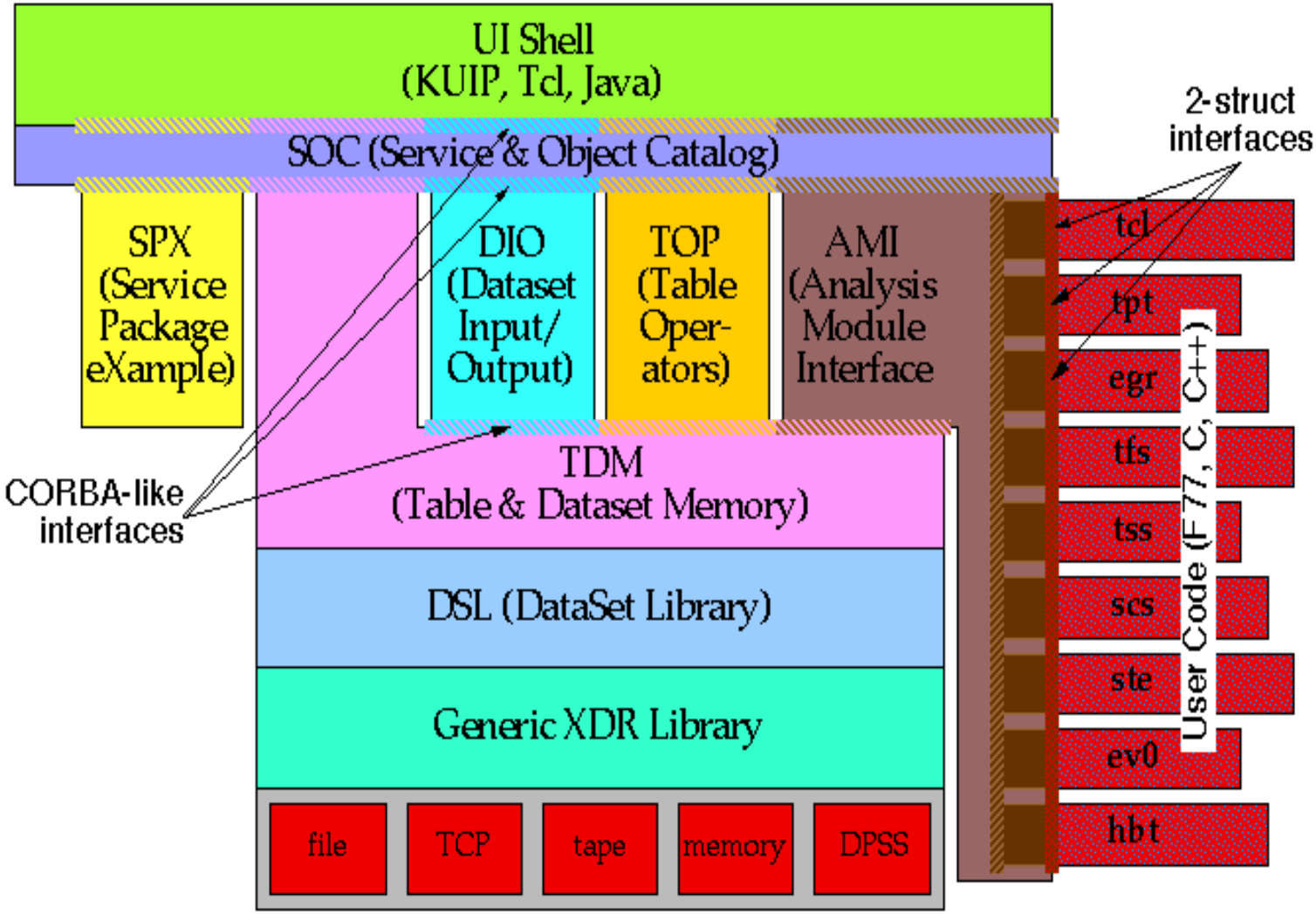
- Horizontal AND Vertical Modularity
- TRUE Component Software
  - —Components go in AND out
- Explicit graceful retirement
- Adoption or imitation of industry standards
- Scripting access to underlying code API
- Reliance on code & doc generation tools
  - —stic, SWIG, idldoc, stafgen, pamigen, Orbix idl
- Dynamic Linking & Unlinking of Components
- Multi-language support
  - —Legacy software (& larger FTE pool)
    - N.B. C++ will be a legacy language before 2015

Software Bus & Modularity



- "Horizontal Mod." reflects ASP domains (ie. functionality).
- "Vertical Mod." reflects protocol & interface standards.
- CORBA provides a well defined interface standard, a fully functional software bus, & auto. Client-Server capability.
- SOC - Collocated Software Bus ASP
  - — Gateway to real ORB

StAF Architecture



CORBA

- Non-Proprietary Software Bus Specification
  - —Object Oriented Client-Server Architecture
- Distributed Computing
  - —Location Transparency
  - —Heterogeneous, distributed environment
  - —Event Granularity Þ Event Farming
- Information flow between processes
  - —Data Table - large & simple (XDR-opaque)
  - —Control Messages - small & complex
- IDL - Interface Definition Language
  - —Purely descriptive -- No operations
  - —Syntax & scoping similar to C++

STAF Objects & Packages

- ASP - Object Factories (singletons)
  - —create/find/delete worker objects
- ASP - Worker Objects
  - —C++ classes to do system-like work of FW
- PAM - Calculation Objects
  - —C++ classes for user-written algorithmic code
- Data Objects
  - —Tables: smart structs (run-time type checking, etc.)
  - —Datasets: containers (UNIX FS-like hierarchy)
- Error Stack - ASPs & PAMs condition values
- Result Stack - queried attribute values & scratch

Analysis Service Package

- CORBA-compliant C++ class library
  - —1 Object Factory Class (singleton)
  - —0 or more Worker Object Classes
- Communication & Control – Software Bus
- Serves a single purpose or set of purposes
  - —System-like Services (I/O, Mem mgt, DB, etc.)
- Normally independent of other ASPs
  - —Limited interdependency realizable
- Can be statically or dynamically linked
- Generic - not specific to data table types
- Directly available from UI, CORBA, or code

Object Factories

- **Factory Attributes/Methods**

```
readonly attribute long count;
  - Count of manufactured objects
STAFCV_T list();
  - Method to list manufactured objects
```

- **Methods for each worker class (eg. thing)**

```
thing* newThing(in TYPE ARGUMENT, ...);
  - Construct a new thing object
thing* findThing(in string SORef);
  - Find the thing object specified by SORef
STAFCV_T deleteThing(in string SORef);
  - Destruct the thing object specified by
  SORef (Stringified Object Reference)
```

IDL in StAF

- defining table types

IDL	Ansi-C/C++	FORTRAN
short	short	INTEGER*2
long	long	INTEGER*4
char	char	CHARACTER*1
float	float	REAL*4

  —scalars, vectors, arrays, structs
- defining analysis module interfaces (type 1 API)
  - `interface`
    - Define analysis module interface
  - `in, out, inout`
    - Define I/O mode for table arguments to module
- defining service package interfaces

  —Fuller subset of IDL usable

stic - STAR IDL Compiler

- Automates many user tasks
  - —Eases learning curve
  - —Enforces consistency w/ architecture
  - —Speeds development
- Dataset Tables
  - —Automatic generation of .h & .inc files
- Physics Analysis Modules
  - —Automatic generation of interface code
  - —Generation of PAM skeleton code
  - —Automatic generation of life-cycle code

stic - usage

- **lex & yacc based IDL compiler - H.Ward**

```
Usage: ./stic [-h?rMTivftq] [-Iincdir] [-static|-dynamic] [xxx.idl]

-? Prints this usage message and then immediately quit.
-dynamic Dynamic tables.
-f Produce only header files (.h and .inc).
-h Prints this usage message and then immediately quit.
-H Produce only the header   files.
-i Ignore case (upper converted to lower).
-I Mechanism for specifying list of include directories.
-M Write string to stdout for use in a Makefile, no other output.
-T Write string with used tables to stdout for use in a Makefile, no
   other output.
-q Operate quietly.
-static  Static tables.
-t Produce only the template files.
-r Produce only the ROOT files.
-v Write version info to stdout, no other output is produced.
```

User Code - Analysis Modules

- **User-written physics code (F77, C, C++)**
- **Interface defined by IDL**
  - `interface findv:amiModule{`
    `STAFCV_T call(inout part p, out event ev);};`
- **Callable from user interface**
  - `KUIP> MODULE/CALL findv part evt`
- **Access to data as program variables**

```
INTEGER*4 FUNCTION FINDV(P_H, P, EV_H, EV)
DS2REALLOCTABLE(EV_H,EV,P_H.NOK)   ! reallocate
table memory
EV_H.NOK = 1
EV(1).MULTIPLICITY = P_H.NOK
DO I=1,P_H.NOK
 EV(1).MAX_Z = MAX(P(I).Z,EV(1).MAX_Z)
ENDDO
```

DSL - Datasets & Tables

- **Self-Describing Machine Independent Data Format based on Sun's XDR & CORBA IDL**
  - **—disk, tape, network, other**
  - **—AIX, HP-UX, IRIX, Linux, OSF1, SunOS 4.x, Solaris 2.x, Solaris X86, CRAY T3E, Windows**
- **Complex Data Types & Hierarchic Datasets**
  - **—directed acyclic graphs**
- **Low overhead & high performance**
- **Dynamic Memory Allocation**
- **Random Access (Table = atomic data unit)**
- **C++ Class Library (100% consistent with C API)**
- **RDB Functions (Join, Project)**

Unix-like Dataset Hierarchy

- **SAMPLE**

```
**************************************************
EVENT---+CONFIG-+Run                                *
        |       |                                   *
        |       +CALIB---...                        *
        |                                           *
        +RAW----+TPC----+SECT01-+Sequence           *
        |       |       |       |                   *
        |       |       |       +Data               *
        |       |       |                           *
        |       |       ...                         *
        |       |       |                           *
        |       |       +SECT24-+Sequence           *
        |       |               |                   *
        |       |               +Data                *
**************************************************
```

- **Unix-like commands.**
  - **cd, cp, ln, ls, mkdir, mv, pwd, rm, rmdir**
- **Data are logically organized by hierarchy.**

StAF User Interface

- **Interface layered for "Graceful Retirement"**
- **Scripting Language vs. GUI**
  - **—More control/programmability (GUI second)**
- **Current interface implemented by KUIP**
  - **—Fully functional Tcl interface implemented - DP**
- **Interface reflects underlying classes**
  - **—Maximum control at user interface level**
  - **—Less transition penalty between languages**

C++
```
dioFileStream * oldat = dio->newFileStream("oldat","old_data.xdf","R");
oldat->getEvent(".");
```

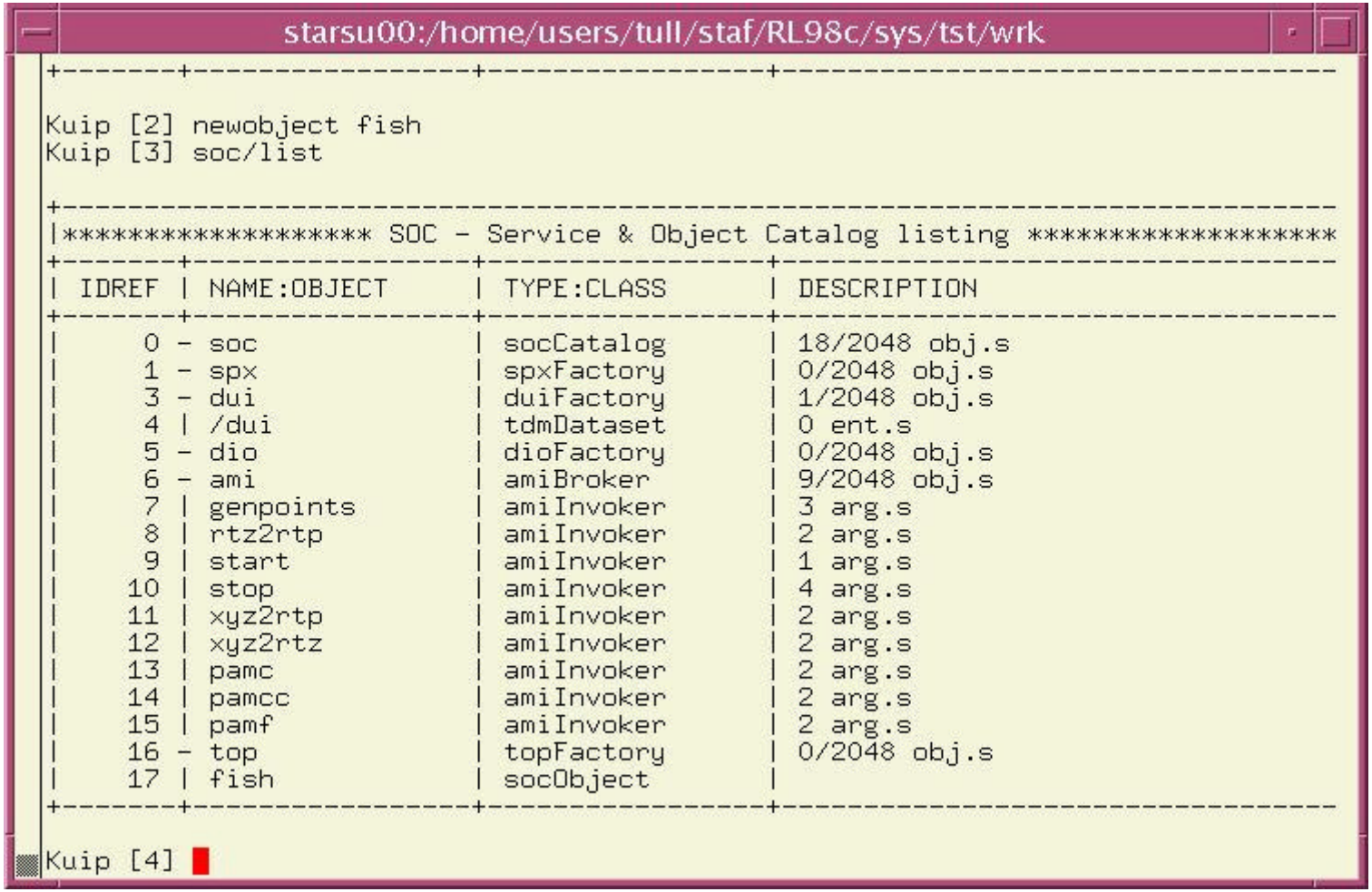
KUIP
```
DIO/NEWFILESTREAM oldat old_data.xdf R
GETEVENT olddat .
```

Tcl
```
dio newFileStream oldat old_data.xdf R
oldat getEvent .
```

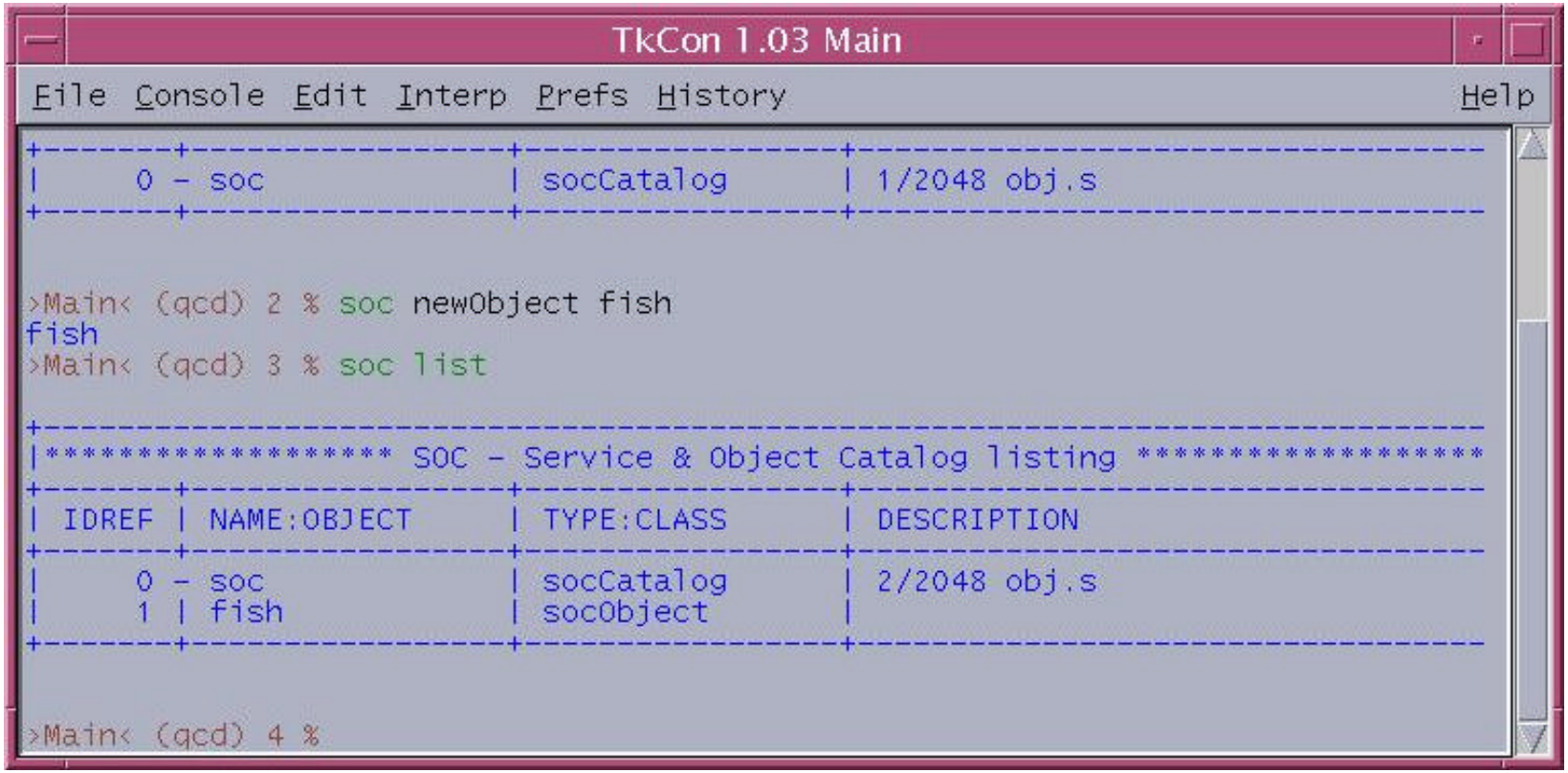
The right tool for the job

- **Programming languages are designed for building data struct.s & algorithms from scratch.**
  - **—strongly typed to manage complexity**
- **Scripting languages are designed for gluing applications.**
  - **—higher level of programming**
  - **—more rapid application development**
    - **programmers write same # of LOC per year**
  - **—programmable applications (User Interface)**
- **trade off execution efficiency for devel't speed**
  - **—computers are getting faster**
  - **—FORTRAN & ALGOL vs Assembler in early 60s**

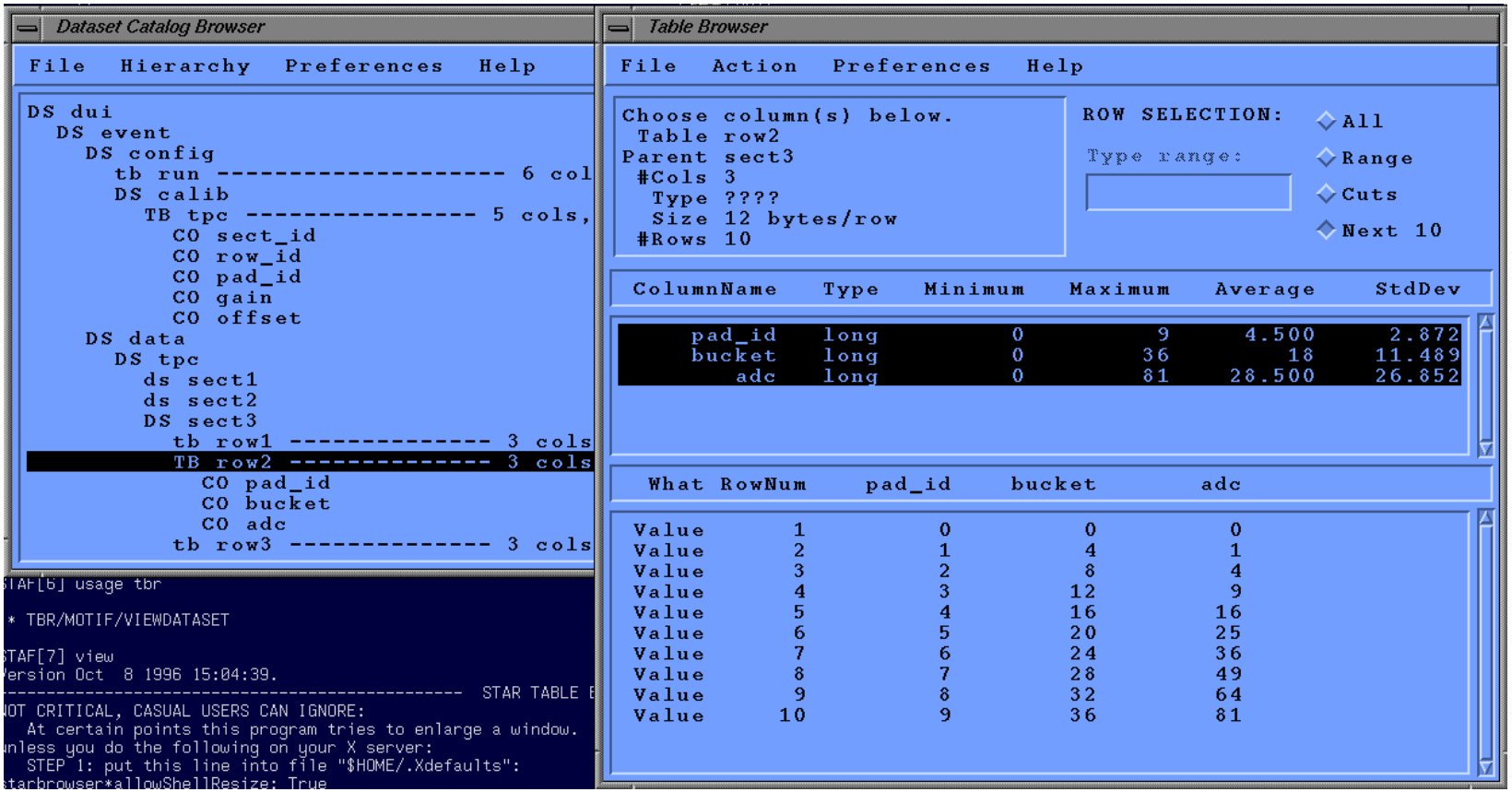
KUIP StAF Interface



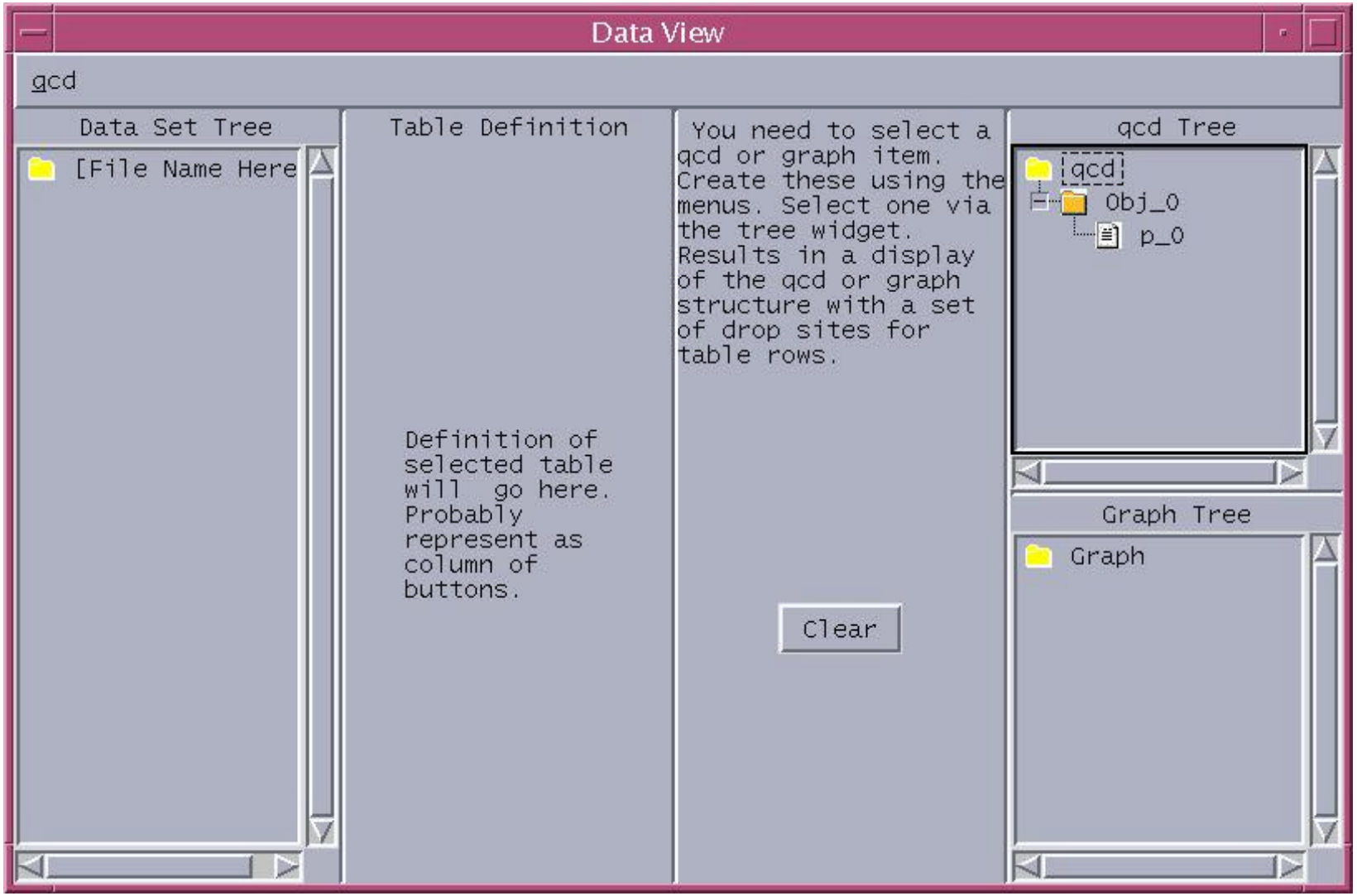
Tcl StAF Interface



StAF Table Broswer



Tcl/Tk BLT Tree Tool



StAF ASPs & PAMs

- AMI - Analysis Module Invoker
- DIO - Dataset Input/Output
- DUI - Dataset Unix-like Interface
- EML - Error Messaging & Logging
- GCA - Grand Challenge Archictecture
- ISS - DPSS Input/Output
- LEV - Logging of Environment & Versions
- NIO - Ntuple Input/Output
- OIO - Objectivity Input/Output (non-functional)
- PIO - PHENIX Input/Output
- RDB - Relational (Oracle) Database
- RTI - Root Table Interface
- SOC - Service & Object Catalog
- SPX - Service Package eXample
- TBR - Motif Table BRowser
- TDM - Table & Dataset Memory
- TNT - Table to NTuple
- TOP - Table RDB OPerators

- STAR Analysis pre-MDC1
  - 400 Tables
  - 150 PAMs

StAF OS Coverage

- **StAF has successfully built and run on:**
  - **—AIX, HP-UX, IRIX, Linux, OSF1, SunOS 4.x, Solaris 2.x, Solaris X86, CRAY T3E, Windows NT 4, Windows 95**
- **Current state of makefiles**
  - **—AIX, IRIX, Linux, Solaris 2.x (egcs, CC 4.2, CC 5.0 w/ -compat=4)**

Projects Using StAF

- **STAR**
  - **—Most physics SW development**
  - **—Successful MDC1**
- **PHENIX**
  - **—MDC1 & MDC2**
- **E896 (AGS)**
  - **—1.4 M real events analyzed**
- **Grand Challenge**
  - **—First development & deployment platform**
- **Clipper**
  - **—Wide Area Scientific Data Access project**
  - **—35 MB/sec I/O between SLAC & LBNL**

ATLAS - Analysis Framework

- **Data Analysis Framework-"like" Programs**
  - **—ac++, arve, basis, cleo, d0 fw, dspack, gaudi, jas, lulu, openscientist, paw, root, staf, tas, ...**
- **Publicly Available Tools & Technologies**
  - **—ace, coool, corba, ddl, grdoc, idl, idldoc, ilu, java, javacc, jini, mysql, netlogger, odl/oif, python, sun idl-cfe, swig, tcl, tk, uml/xmi, ...**

- **ATLAS Framework must be in place EARLY (now) & LONG (~20 years!).**
  - **—Challenge: Allow adoption of technologies unknown at time of 1st version of FW.**


BERKELEY LAB

SLAC Þ LBL -- DPSS Write Test

SLAC Ü LBL -- DPSS Read Test