Development of a new VLBI data analysis software

Sergei Bolotin, John M. Gipson, Daniel S. MacMillan

NVI, Inc.
7257D Hanover Parkway
Greenbelt, MD 20770
NASA Goddard Space Flight Center
Greenbelt, Maryland 20771 USA

6^{th} IVS General Meeting
Hobart, Australia, February 7-13, 2010

Overview

1. Introduction
2. Conceptualization: What we do want from a new version?
   - Goals
   - Core requirements
3. Analysis: What should the system do?
   - Overview of the system
   - Scenarios of system's behavior
4. Architecture design: How do we develop the system?
   - Architecture overview
   - Common tactical policies
   - Roadmap

VLBI data analysis software

The current CALC/SOLVE system

- The current VLBI data analysis software CALC/SOLVE has been developing since the mid seventies of the last century.
- It is the most widely used data analysis software in the geodetic/astrometric VLBI community.
- CALC/SOLVE software is used in data preparation of VLBI observations. Output of CALC/SOLVE is input to other VLBI data analysis packages.
- The current CALC/SOLVE system consists of more than hundred individual programs, that are written mostly in FORTRAN-90.
- Many significant results were obtained with CALC/SOLVE in various branches of astronomy, geodesy and geophysics.

VLBI data analysis software

The current CALC/SOLVE system: limitations and risks

- The software package was developed as a set of utilities around the Mark-III database access library which was created at the end of 1970's and then has been frozen.
- Since CALC/SOLVE is a production system in VLBI data flow (preparing **Version 4** databases and **NGS** files), adding new models and features was usually done as patches over existing code or even as tricks to keep the existing general structure of the software.
- The database access library and procedures that use it were designed with the computer hardware limitations of the late 1970's.
- As a result, it is hard to maintain and extend the current CALC/SOLVE system.
- VLBI2010 will demand flexibility and performance that the current realization of CALC/SOLVE can not satisfy.

Developing of the new VLBI data analysis software

New software developing

- **NASA GSFC** in 2007 started the process of new VLBI data analysis software development. With the help of the geodetic VLBI community new formats for data storage have been elaborated.
- **NetCDF** has been chosen for binary data storage, the development of file specifications has been finished, and in July 2009 the first conversion from Mark-III database to NetCDF files was performed.
- The process of **software developing** started in August 2009. An *evolutionary* model for software production process was selected. The process consists of the following overlapped in time activities:
  - **Conceptualization**: Formulation of a statement of the problem and establishing core requirements.
  - **Analysis**: Developing a model of the system's behavior. Describing functionality and performance of the system and necessary resources.
  - **Design**: Creatign the architecture for the evolving implementation and establishing common tactical policies.
  - **Evolution**: Growing and changing implementation through successive refinement, leading to the production system.
  - **Maintenance**: Managing post-delivery evolution.

Goals of developing a new version of the software:

The VLBI data analysis software is a **tool** for study of Earth rotation, construction of celestial and terrestrial reference frames and investigation of various geophysical phenomena:

- **Reliable results, reliable software behavior**: The output produced by the software is used by many scientists in the world, we have to be confident in obtained results.
- **Easy to maintain, develop and extend**: It should be easy to add new models, estimated parameters or even types of observables. Further improvement of the software should not force change of its general design.

VLBI2010 system will generate many more observations:

- **Performance issues**: The software should be efficient enough to process observations from the VLBI2010 system and/or from other techniques of modern geodesy in reasonable time. User efforts in processing observations should be minimal.

Scientists work in groups:

- **Data integrity and a group work**: It is supposed that data analysis is performed by a group. The results of the work of one user should be implicitly (transparently) available to the whole group.

Goals of developing a new version of the software:

The following additional targets could be easily achieved when the software is mature:

- Add **data processing of observations from other space geodetic techniques**: Combination with SLR, GPS.
- Implement of **different approaches of estimating parameters**: use LSQ, SRIF, Kalman filter, etc.
- Support of efficient **interaction with other software**: Geodyn, sked, AIPS, etc.
- Support **simulation** analysis.
- Support of **correlator software** maintenance.

Core Requirements: Reliability

Reliable software behavior, reliable results:

To reach this goal the software should possess the following features:

- Quality control:
  - Test suites that check all essential parts of the software
  - Multilevel logging system
  - Error reporting using different delivery methods (write to a file, send a mail)
- Support versioning of input data and solutions
- Ability to reproduce the current CALC/SOLVE results and historical models
- Protect user from making blunders
- Ability to do automated feedback to stations, correlators
- Data visualization:
  - Modern graphical user interface
  - Elaborated plotting system with ability of data editing
  - "Drill down": get info about data down to fringes

Core Requirements: Software developing practice

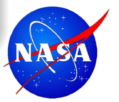
Easy to maintain, develop and extend:

The software should be:

- Modular
- Flexible
- Portable

The process of software developing should:

- Be performed using standard software development tools
- Be a team-oriented work
- Support multiple versions of the software
- Maintain a multilevel documentation system:
  - Comments in sources
  - External memos, flow charts, diagrams, etc.
  - References to books, papers



Core Requirements: Performance

Performance issues:

Minimizing user time:

- Automation
- Script language
- Communication with external source of data using already adopted file formats and standard protocols

Minimizing CPU time:

- Compile-time controlled Debug and Production mode
- Parallelizable:
  - Multi-core CPU: OpenMP API
  - Multi-CPU systems: Threads
  - Computer clusters: Parallel Virtual Machine (PVM), Message Passing Interface (MPI)

In general, software should be able to process observations from fully deployed network of VLBI2010 station in reasonable time.

Core Requirements: Group Work

Data integrity and a group work:

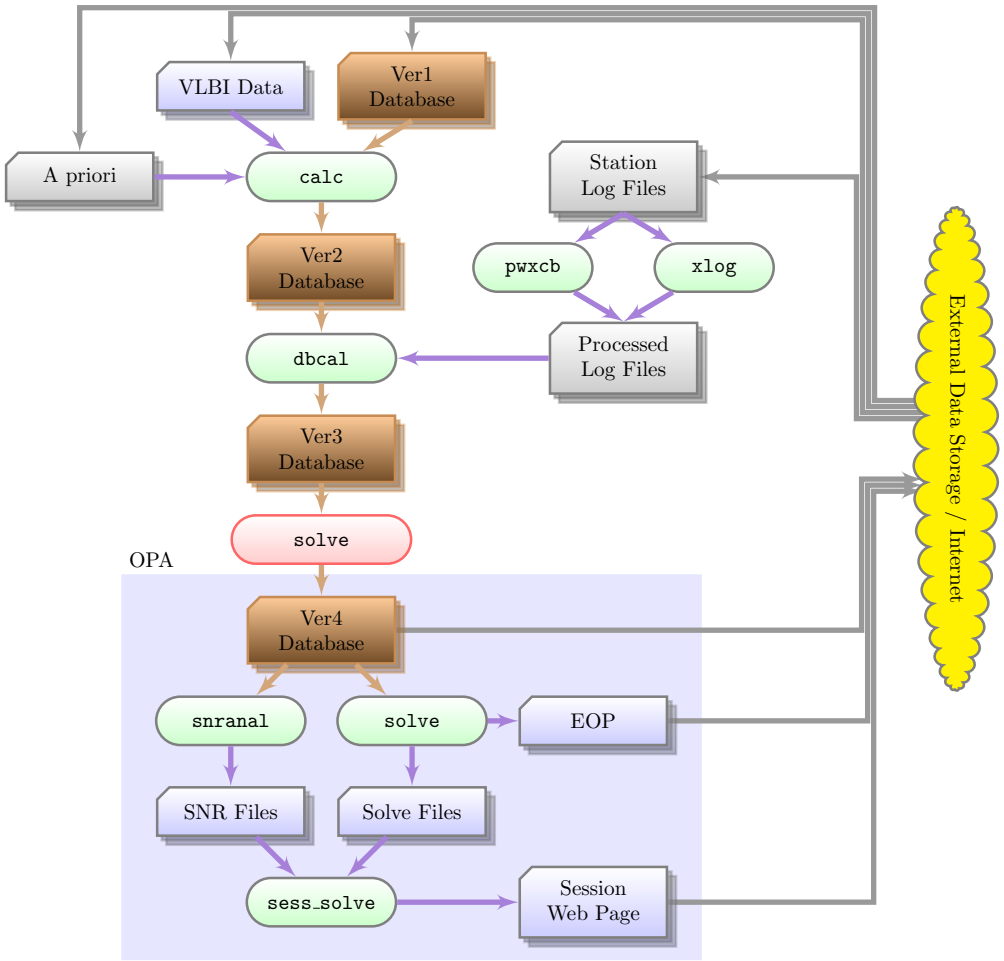
- Data share and multiuser environment:
  - File locking mechanism
  - User administrating
  - Different user levels
- Resources monitoring:
  - Available system resources
  - Resources consumed by each user

Overview of the system

General characteristics

- **Type of software**: Data analysis. This is not a time critical application. As with many similar data analysis softwares, it should read observations in predefined format(s), evaluate theoretical values corresponding to the observables and with the Least Square Method estimate parameters that have influence on the observable values.
- **Hardware**: There are no special design requirements of the hardware. The system could run on a conventional "personal computer" as well as on a dedicated workstations or even a hardware cluster.
- **Operation System**: There is no specific requirement to operation system. We assume POSIX compliance; this could be one of the Linux distributives, Solaris or FreeBSD. The software should be portable to these families of OS.
- **Available Prototypes**: There are two prototypes which can be used in the development process of the new software. We have the current **CALC/SOLVE** system which performs the same tasks. Also we have **SteelBreeze** that could be used for prototyping the Object-Oriented Design and graphical user interface. In general, all proven-by-time algorithms and approaches realized in these prototypes should be used.

Overview of the system: Data Flow, the current realization



Data:

- Input data in various known formats (ASCII, binary)
- MarkIII DBH format
- Input/Output files (ASCII), formats are maintained by IERS, IVS, IGS, other organizations

Procedures:

- Batch running executables
- Interactive executables

Data exchange:

- DBH functions calls
- Format changing data flow
- Copying files / network transfer operations

Diagram info:

Version: 0.1-pre

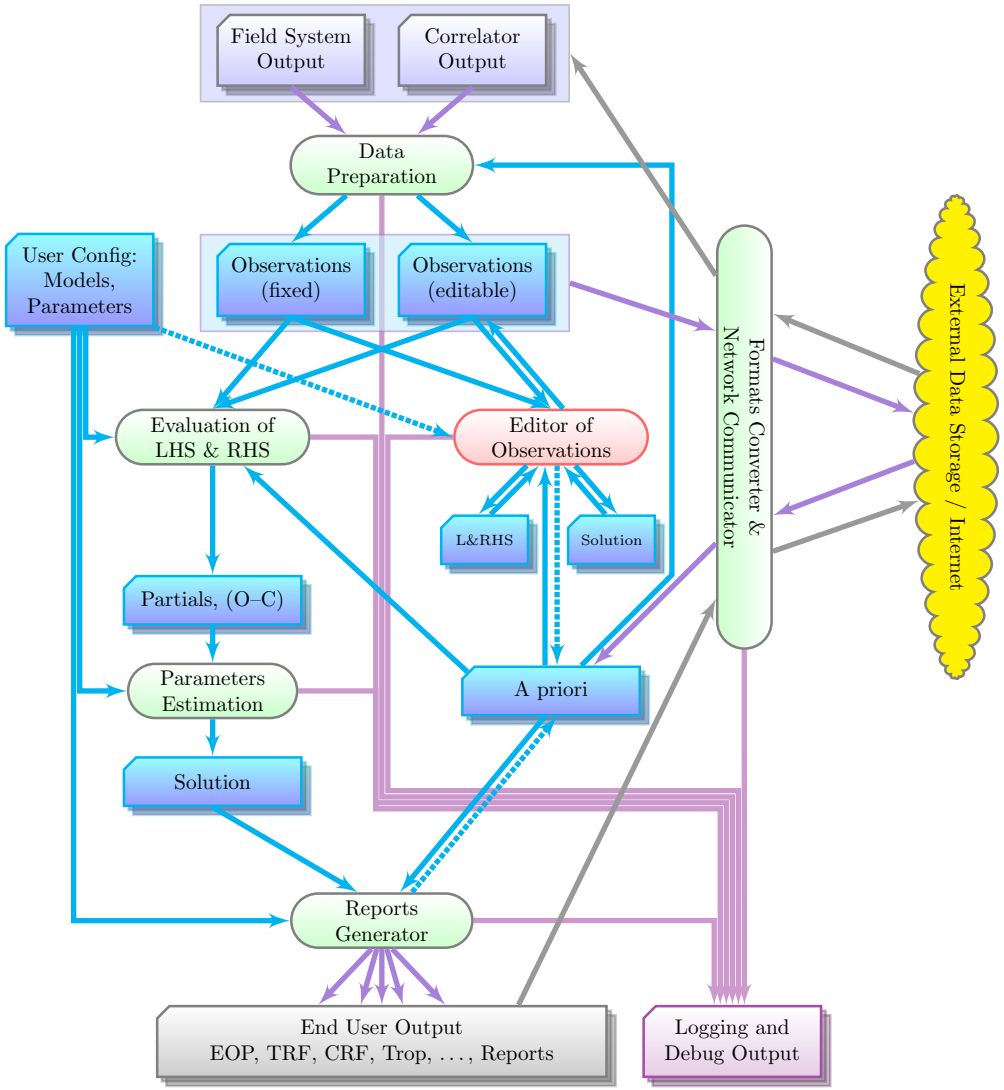
Date: October 14, 2009

Overview of the system: Data Flow

Proposed changes:

- The data files would be physically separated in two parts: observations and editables. Thit could prevent observations from unintentional modification by user.
- Mark-III database handler format will be replaced by open source Net-CDF format.
- Several executables that are used in data preparation, `calc`, `pwxcb`, `xlog` and `dbcal`, will be replaced by one executable block.
- The executable block, **Editor of Observations**, is intended to interact with a user, while other blocks run mostly in a batch mode.
- On the other hand, the block `solve` will be split into several independent parts for evaluation (O-C) and partials, parameter estimation, and further handling of estimated parameters (writing reports, updating *a priori*, etc.).
- The format conversion filters and network transfer utilities are presented in a separate block.
- The logging facilities with common interface are shown on the diagram.

Overview of the system: Data Flow, proposition



Data:

- Input data in various known formats (ASCII, binary)
- Internal data, NetCDF format
- Output files (ASCII), formats are maintained by IERS, IVS, IGS, other organizations
- Logging and debugging output

Procedures:

- Batch running executables
- Interactive executables

Data exchange:

- NetCDF functions calls
- Format changing data flow
- Copying files / network transfer operations
- Log calls

Diagram info:

Version:	0.2
Date:	October 26, 2009

Functionality of the system

Key Function Points:

1. Import observations into local storage. Check reliability of imported data and correct them if it is necessary and possible.
2. Import *a priori* data into local storage, check them, export them in the appropriate formats.
3. Prepare newly imported observations for the analysis, interactive mode. Check for ambiguities, outliers, clock breaks, new sources/stations, anomalous cable calibration data; performing reweightening, etc.
4. Data editing.
5. Multi-session analysis:
   1. Evaluate (O-C) and partial derivatives according predefined set of models;
   2. Form equation systems;
   3. Obtain a solution.
6. Generate reports; update files with *a priori* data (if necessary).
7. Send reports to:
   1. analysts;
   2. outer world.
8. Export checked and validated observations in appropriate formats.

Functionality of the system

System Function Points:

1. Report an error.
2. Send and receive a file through a network connection.
3. Parse a script for a batch run.
4. Check data integrity.
5. Monitor available resources (CPU, memory, hard disk space, network connections, open file descriptors).
6. Lock a file before writing, unlock it after changes finished.

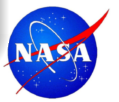
Programming language

Programming language:

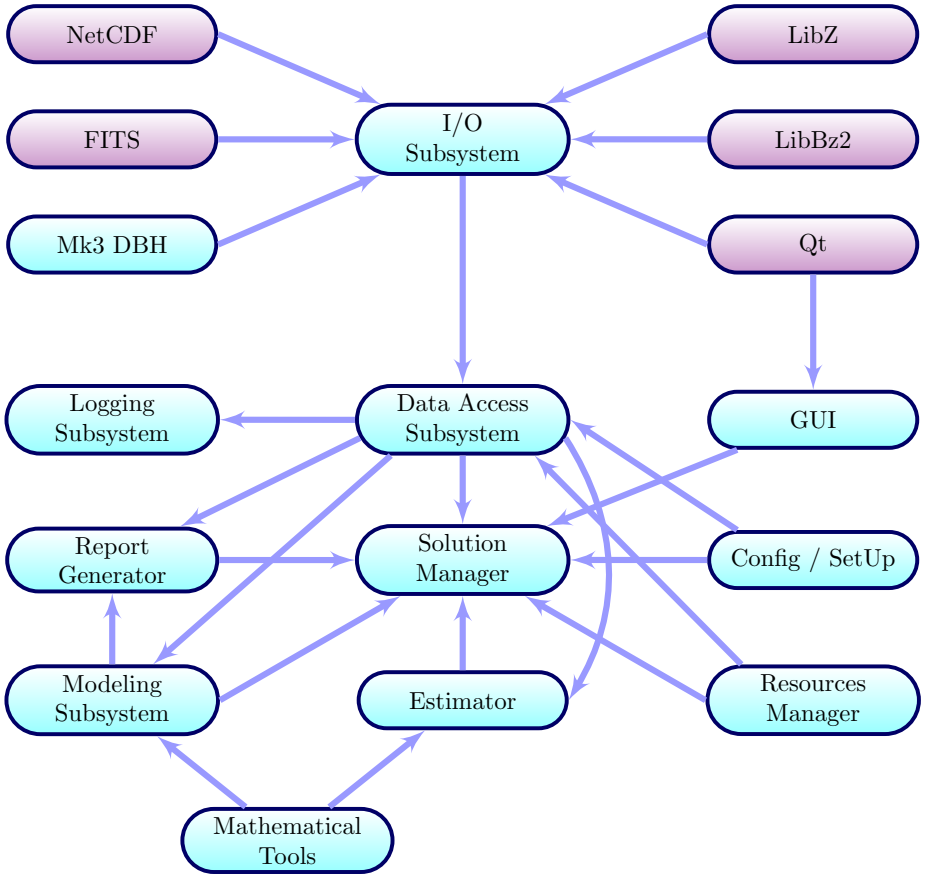
Choosing a programming language is an important decision. The language should be **"natural"** to the task. For example, *division of one number by another one is a routine application using Arabic numerals, but it is a nontrivial task when the Roman numerals are used*.

Advantages of C++:

- This type of software (data analysis) can be easily and efficiently described with the Object Oriented Programming language.
- C++ is powerful, flexible and efficient. It supports (in addition to arrays) various data structures like lists, trees, queues, collections, maps, etc., as well as user defined types.
- The current realizations of C++ have already achieved maturity and are used in many business and scientific projects.
- Most external packages, like GUI or network access libraries, are written in C/C++.



System Decomposition



Modules:

Software Subsystem

External Library

Dependencies:

Data structures, constants, function calls

Comments: The Logging Subsystem provides its interface to all local subsystems

Diagram info:

Version: 0.2

Date: February 1, 2010

Organization of the software

Shared library

Code reuse can significantly improve performance of software development. All essential procedures should be organized in one or several libraries, so they will be accessible for other projects. Executable binaries should be "drivers" that use the procedures from the library.

The library should provide data structures of various objects, procedures to manipulate them and tools – additional general components that are necessary to implement a proper realization of the library.

Using the library could improve productivity not only in VLBI data analysis, but in the whole domain applications. E.g., the utility for comparison source catalogs would use library's data structures that represent a source and a catalog, and procedures that implement reading files and the LS estimation.

One of the benefits of such realization is the ability to update the code without recompiling executable binaries – only a change of the shared library will be necessary.

Organization of the software

Example of using the library

The sources of a utility that converts observations from Mark3 DBH format to NetCDF could looks like:

```
int main(int ARGC, char *ARGV[])
{
 /* [check paths, existing files, ability to write, etc.] */

 // constructors:
 int          RC;              //return code
 dbh_stream   DBStream(...);   //an object that reads/writes a file in Mark3 DBH format
 nc_stream    NCStream(...);   //an object that reads/writes files in NetCDF format
 VLBIsession  Session(...);    //an object that stores VLBI observations and auxiliary data

 // read a file in Mk3 DBH format:
 DBStream >> Session;

 // check session data integrity, if it is OK make an output:
 if (Session.selfCheck())
    Session >> NCStream;

 /* [check status of the output operation, evaluate and return RC] */
 return RC;
}
```

All real work is done in the library.

Scalability

The new VLBI software and VLBI2010

One of the major challenges that will arise with emerging the VLBI2010 system is a huge increase in number of observations:

$$3 - 6 \text{ thousands of observations} \Longrightarrow \text{hundredth of thousands of observation}$$

- **Storage**: Modern hard drives are cheap enough. To prevent data loss due to hardware failure, we could organize data storage in RAID or set up a dedicated file server.
- **CPU time**: Due to the nature of the task, we can increase the performance by paralellizing data processing.

  Now there are two schemes to run a task in parallel:

  1. **OpenMP** or **threads** – lightweighted parallel processes that share the same data. These paralellizations can be realized on a single computer with a multi-core CPU or several CPUs.
  2. **computer cluster** – a dedicated cluster of computers that communicate with a special network protocol (e.g., MPI, PVM).

  The software should be thread-safe and ready for modifying most time consuming parts to execute in parallel.

Common tactical policies:

Common tactical policies:

- Using common **programming style convention** increases readability of the source codes for members of the development group. We can adopt any of the existing style conventions. It should cover:
  - indentation;
  - types naming convention;
  - constants and variables naming convention;
  - procedures naming convention;
- Applying a **test suite** for every essential part of the system assures software quality at the development stage and during distribution to external users.
- For portability issues we could use **autogen/autoconf** suites, which are now standard de facto in the open source community. Proper use of these suites allows efficient porting software to a new operation system.
- Automatic generation of reference documentation. With **Doxygen** utility it is possible to generate reference manuals from properly formatted comments.
- **Version control system** will support simultaneous access to different versions of the software.
- Automatic **backup** of the source tree will prevent hardware failure.

Roadmap:

Tentative schedule

Feb 2010: The conception of the new software will be presented to the geodetic VLBI community on the 6^{th} IVS General Meeting. We can expect some additions and corrections to the proposed design.

Mar 2010: Revision of the whole system.

Apr 2010: An **Evolution stage** of the software developing process should begin in the 2^{nd} quarter.

Jul 2010: First release of interactive version of νSolve should be available.

Oct 2010: Integration of current `pwxcb`, `xlog` and `dbcal` functionality into νSolve.

Jan 2011: Validation tests and comparison with the current CALC/SOLVE system for the interactive νSolve should be finished.

Comments and suggestions are welcome