DEVELOPING A MULTI-GNSS ANALYSIS SOFTWARE FOR SCIENTIFIC PURPOSES

Michael Meindl[(1)], Rolf Dach[(1)], Stefan Schaer[(2)], Urs Hugentobler[(3)], Gerhard Beutler[(1)]

[(1)]Astronomical Institute, University of Bern, Sidlerstrasse 5, 3012 Bern, Switzerland, Email: meindl@aiub.unibe.ch
[(2)]Swiss Federal Office of Topography swisstopo, Seftigenstr. 264, P.O. box, 3084 Wabern, Switzerland
[(3)]Technische Universität München, Arcisstr. 21, 80333 München, Germany

ABSTRACT

Upcoming next-generation Global Navigation Satellite Systems (GNSS) and planned updates of existing systems call for multi-GNSS capable analysis tools.

The development or extensive update of a scientific multi-GNSS software is a demanding task. In a first step a list of requirements is compiled, taking into account not only scientific but also technical needs. In the following conceptual phase the fundamental software architecture and a set of basic design principles are specified. The last step is the actual programming of the software.

The update of the Bernese GPS Software (BSW) to full multi-GNSS capability is presented as real life example for realizing the design principles in a software package.

1. INTRODUCTION

To scientifically profit the most from the availability of different Global Navigation Satellite Systems (GNSS), a combined and fully consistent analysis of tracking data from all systems is essential.

This approach is already followed at the Center of Orbit Determination in Europe[1] (CODE) since May 2003 for the analysis of data from the American Global Positioning System (GPS) and the Russian Global Navigation Satellite System (GLONASS). CODE is one of the International GNSS Service (IGS) Analysis Centers (AC). Reference [1] is a detailed status report on CODE's GNSS processing. All computations at CODE are performed using the Bernese GPS Software (BSW).

The Bernese GPS Software is a scientific analysis tool for GNSS data and is developed at the Astronomical Institute of the University of Bern (AIUB). In its present version [2] the BSW can process two-frequency carrier phase and pseudorange observations from GPS, as well as GLONASS. Satellite laser ranges can be analyzed, too, but are not further considered in the scope of this work.

The planned inclusion of upcoming GNSS (e.g., the European Galileo system) in the BSW has given cause for an outright redesign of the software package.

A list of scientific, as well as technical requirements for a multi-GNSS software is compiled. Subsequently, the fundamental software architecture and basic design principles and concepts are defined. The design is not only dictated by the list of needs, but also by available infrastructure such as programming language.

The update of the BSW to full multi-GNSS capability is presented as example for developing a scientific multi-GNSS software package. The implications of realizing the defined design principles are presented.

2. REQUIREMENTS FOR A MULTI-GNSS ANALYSIS SOFTWARE

The first step in developing (or expanding an existing package to) a multi-GNSS ready scientific analysis software is to define a set of requirements. New features offered by upcoming or modernized GNSS do not only demand for scientific but for technical requirements, too. The list of requirement is not static but subject to change during the development process. However, the most important points are considered and presented in the following sections.

2.1. Scientific Requirements

A quite obvious but nevertheless important innovation offered by new and updated GNSS is transmitting navigation signals on more than two frequencies (as is the case now for GPS and GLONASS). Established linear combinations (LC) may be formed with varying frequencies and a multitude of new LCs with more than two components may be formed. These possibilities are interesting for, e.g., data pre-processing, ambiguity resolution, and parameter estimation.

[1] CODE is a joint venture of the Astronomical Institute of the University of Bern (AIUB, Switzerland), the Swiss Federal Office of Topography (swisstopo, Wabern, Switzerland), the Federal Agency for Cartography and Geodesy (BKG, Frankfurt am Main, Germany), and the Institut für Astronomische und Phsyikalische Geodäsie of the Technische Universität München (IAPG/TUM, Munich, Germany).

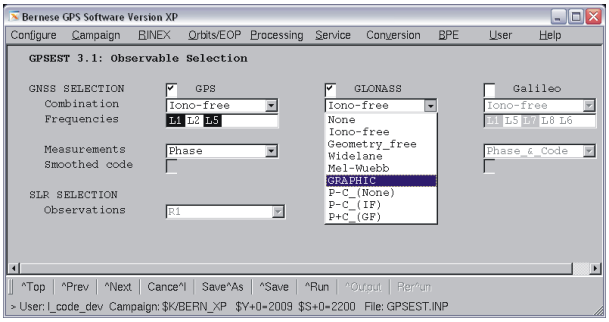

Figure 1: Observable selection in the BSW.

Fig. 1 shows the reworked observable selection panel of the BSW to illustrate the various possibilities to form linear combinations.

Another novelty is offered by exchanging observation data using the Receiver Independent Exchange Format (RINEX) in Version 3.xx as described in [3]. With this format, the tracking technology of receivers finds its way in the data analysis. This is important for the combination of data from receivers of different types and from different manufacturers. But even one and the same receiver may use diverse tracking technologies, e.g. for different GNSS or satellite models. The correct handling of all resulting biases becomes possible.

The maximum number of visible satellites and global coverage is improved. In a combined multi-GNSS analysis the number of available observations increases and may help the estimation of certain parameter types, such as Earth orientation parameters (EOP).

Different orbital characteristics (e.g., revolution period) of additional satellite constellations may help to identify and understand, e.g., periodic variations in time series of estimated parameters. For investigations in this area a GNSS-specific parameter setup is interesting, e.g., to keep the EOP separate for each GNSS and analyse differences between the spectra in resulting time series.

A lot of new applications and possibilities will be found in future, some of them yet unforeseeable. Nevertheless, several basic requirements can be compiled:

- easy and flexible handling of linear combinations of observations with two or more components,
- consideration of receiver tracking technology types, especially correctly dealing with all resulting biases, such as differential code biases (DCB), inter-system biases, and initial phase biases,
- correct and consistent combination of observation data from single- and multi-GNSS receivers,
- single- or multi-GNSS analysis in all possible combinations (GPS-GLONASS combined, Galileo-GLONASS combined, Galileo only, etc.) without preferences for a specific system,
- fully consistent analysis of all involved GNSS on observation as well as on normal equation level.
- offer extensive statistics for each parameter type (e.g., contributing GNSS) to allow studies of the influence of GNSS-specific frequencies on resulting time series (e.g., the sidereal day in GPS derived time series as for instance reported by [4]).

The above list offers a lot of interesting fields of scientific research. It represents the minimum needs for a competitive scientific multi-GNSS analysis software.

2.2. Technical Requirements

Besides the highlighted scientific aspects, the inclusion of new GNSS and updated systems entails a number of technical problems, too. These are mainly caused by two things.

Any modern multi-GNSS software must be open for future developments (which may already be drafted), without the need of a recurrent software redesign. This is especially important for file format definitions.

The second point is a massively increasing number of satellites, observations, and parameters (to name only a few). For example, the inclusion of a fully deployed additional GNSS in the CODE IGS processing would add about 1100 parameters to be estimated[2], raising the total number by about 20% with respect to the current analysis situation using data from GPS and GLONASS (49 satellites) collected by 240 stations. The necessary inclusion of new tracking station will significantly raise these numbers, too. Problems with respect to memory requirements, file size, and computation speed may be follow-up consequences.

To satisfy the technical needs of multi-GNSS capability, the following requirements should be met:

- very flexible and easy expandable file formats,
- easy inclusion of new models and parameter types in the software,
- unproblematic inclusion of new GNSS (like the European Galileo) and system updates (e.g., GPS modernization),
- easy inclusion of additional linear combinations and tracking technologies,
- generic observation type and carrier frequency handling,
- flexible parameter setup (GNSS-specific, satellite-specific, etc.),

[2] 30 satellites with 18 orbit parameters, and 19 antenna parameters, each. Additional introduced ambiguities are not considered in this context.

- maintain an easy expandability of the software with new programs or program parts,
- no hardwired limitations (e.g., number of satellites),
- rigorous dynamic memory management.

Besides the implications on the software architecture, demands are made for the used programming language. An object oriented approach may be preferred but is not imperative. If compatibility with existing code must be maintained (e.g., if new functionality should be implemented in an existing software) the selection of the programming language is limited. However, some basic high-level capabilities like a dynamic memory management are absolutely mandatory.

3. SOFTWARE ARCHITECTURE AND DESIGN PRINCIPLES

Before programming a complex software package (or updating an existing one), the fundamental software architecture and some basic design principles must be specified. Starting point for these considerations are the available infrastructure, programming environment, and in particular the list of requirements as compiled in the previous section.

Fig. 2 shows a conceptual software design for the new BSW realizing high modularization, expandability, and flexibility. The main processing programs should be kept as independent from external factors as possible (indicated by the dashed line). They communicate with the "outside" world only via well defined interface routines (big black arrows). Shared data, specifications, file formats, and definitions are organized and encapsulated in core modules together with the respective functions. External files are always converted to software inherent formats for internal use. The graphical user interface (GUI) is a separate program part and kept independent from the processing programs.

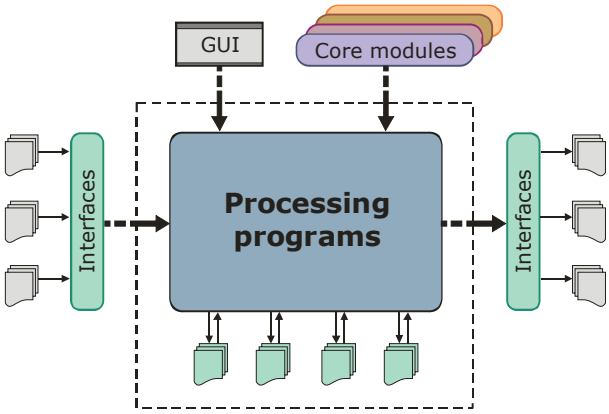

Figure 2: Software design diagram.

Three basic design principles have been found to be especially important to satisfy the list of requirements and to realize the software architecture, namely

- the concept of modularization,
- full independence from external formats, and
- high flexibility and expandability.

These basic principles should be strictly adhered to, thus guaranteeing a comprehensible and consistent software design.

3.1. Modularization

The software should be highly modularized to be able to react on yet unforeseen developments without rewriting the complete software package.

All processing programs depend on various specialized core modules. The processing programs only realize the mathematical models for, e.g., data screening, parameter setup, or normal equation manipulation. In particular, they do not contain any GNSS-specific information whatsoever.

The core modules, however, contain all the shared data, definitions, and information needed by the processing programs to fulfil their respective tasks. This includes GNSS-specific information (such as number and frequency of carriers), receiver-specific information (e.g., available tracking technology, trackable systems), observable definitions (like creation rules for linear combinations), and general information like time system definitions. Fig. 3 gives an overview of the most important core modules of the redesigned BSW.

The processing programs can access the module data only by using dedicated interface routines. These access routines are defined and coded in the corresponding core modules. The programs do not need knowledge of implementation details, only the interfaces must be readily available.

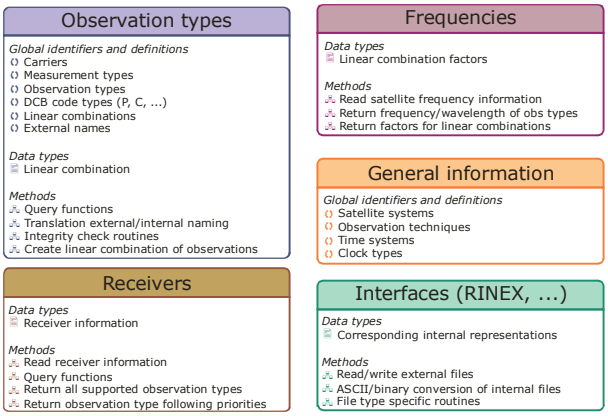

Figure 3: Configuration of the BSW core modules.

This approach greatly facilitates new implementations, especially of new frequencies and linear combinations. As long as the interfaces do not change, only the corresponding core modules must be adapted and the changes are at once available throughout the whole software.

3.2. Independence From External Formats

Another important principle is the independence of the processing programs from external factors. The main programs are using solely software internal file formats and naming conventions. Data is exchanged between single programs via internal file formats, too.

Information from external files (e.g., observation data from RINEX files, Earth orientation parameters) is converted to a software internal file format using a collection of dedicated interface routines (converters). File format specifications, definitions, and conversion, access, and manipulation routines are encapsulated in corresponding modules.

External nomenclature (e.g., the observation codes as defined in the RINEX 3.xx format) is translated by the interface routines (usually during a conversion step) to a set of internal global identifiers. These are defined in the core modules and used in all programs.

In case of changes or updates of foreign file formats or nomenclature, only the corresponding module must be adapted. As long as the interfaces do not change, external file formats do not have any direct effect on the central processing programs.

Another advantage may be seen in the fact that internal file formats may already be prepared for future applications.

Nomenclature for software output files (such as the abbreviations for observation types) are defined in core modules, too. They can be accessed by processing programs only via the interface routines. Thus, a consistent naming for all output files is guaranteed. Should it be necessary to change, e.g., the name of a linear combination, only one module must be changed.

3.3. Flexibility and Expandability

In view of future developments and increasing pace of scientific innovations in the field of GNSS and space geodetic techniques, special attention should be paid to flexibility and expandability of the software package.

There should be no (or at least as few as possible) hardwired limitations such as array sizes, file lengths, or maximum dimensions (e.g., of satellites, stations, etc.) in the software. This calls for a highly dynamic memory management made possible by coding in a modern programming language. Without these hardwired limitations, however, it is left to the developer to foresee a number of security mechanisms to keep the memory usage and computation time in reasonable limits.

As a further consequence, internal file formats and the corresponding interface routines should be as generic, flexible, and expandable as possible. Files are stored in binary format if file size and time critical access play an important role, otherwise ASCII format is preferred. Easy human readability of ASCII files should be maintained wherever possible but not at the expanse of storable information.

Fig. 4 shows the redesigned DCB file of the BSW as one example of a flexible and open file format. It offers a variety of bias types (differential, absolute, relative) for an arbitrary combination of stations, satellites, and systems. Freely selectable code types, flags, and time windows add additional flexibility. The file is kept in ASCII format to maintain readability and facilitate exchange and publication of the contained information.

The graphical user interface of the software is separated from the actual processing programs. A dedicated GUI program displays option panels, reads the user input and prepares information for further usage. The processing programs gain access to user input (e.g., selected files and options) only by a set of input subroutines. These input routines may already validate the user input thus minimizing errors during the actual program run. A redesign of the GUI or implementation of new options does not necessitate changes in the processing programs (of course the programs must eventually be adapted to make use of the new option).

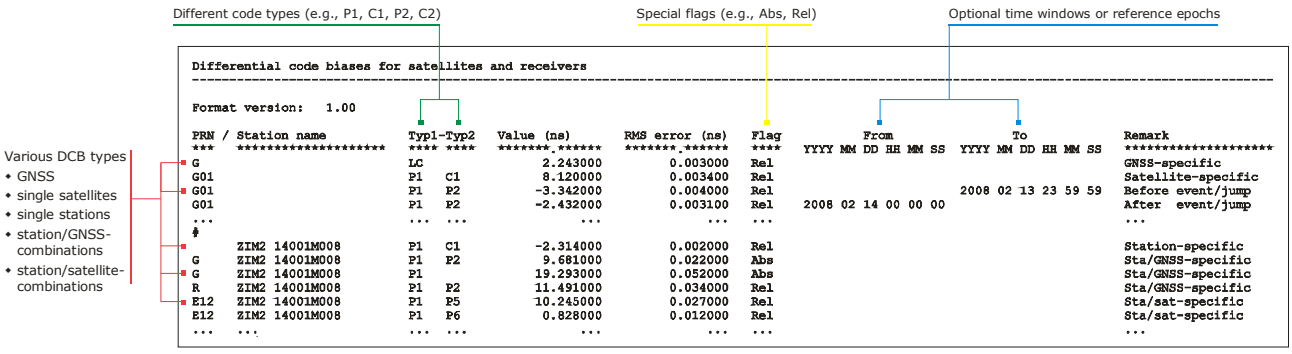

Figure 4: Redesigned DCB file format of the BSW.

4. PRACTICAL EXAMPLE

The implications of realizing the previously presented design principles are demonstrated using the upgrade of the Bernese GPS Software to multi-GNSS capability as real life example.

The Bernese GPS Software is a scientific software tool to analyze GNSS observation data. It was developed at the AIUB back in the mid 1980s and is ever since updated and improved. The basic design, however, was never changed considerably. The current release is Version 5.0 [2]. The BSW was originally coded in FORTRAN-77, since 2000 all developments are solely made in Fortran 90[3]. Fig. 5 shows chronologically the creation dates and programming language of the BSW programs. The solid line gives the total number of programs contained in the BSW package (103 as of September 2009).

The BSW was initially designed to process two-frequency carrier phase and pseudorange data from GPS. In 1999, GLONASS capability was added (cf. [5]) by adapting and updating the already existing software structures and algorithms. Cum grano salis, GLONASS was treated as a "GPS with some peculiarities". This approach was possible, as GLONASS is quite similar to GPS in terms of system design and applicable models and algorithms. A crucial point was the two-frequency design of GLONASS.

Upcoming next-generation GNSS and improvements of existing systems offer more than two frequencies and new observables. In addition, the number of satellites, observations, and parameters to be estimated rises significantly. All these points render a straight-forward implementation, as followed for GLONASS, virtually impossible. A radical software redesign was necessary, following the principles from the previous sections.

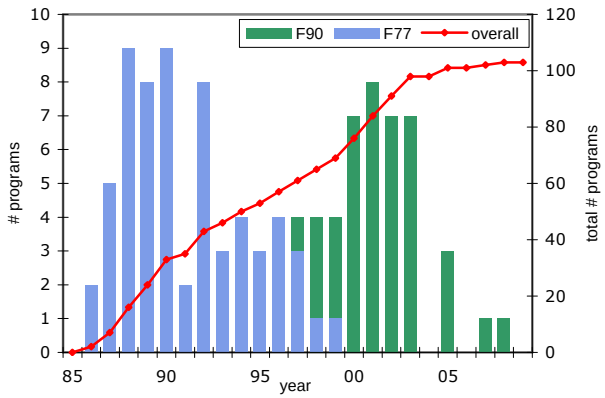

Figure 5: BSW programs creation dates and programming language.

[3] FORTRAN-77 and Fortran 90 are not object oriented programming languages. However, they are widely used within the scientific community.

The update comprises obvious tasks like redesigning and implementing the complete frequency, observation, and linear combination part; redesigning a multitude of file formats, corresponding interface, access, and conversion routines; adapting all programs to the new file formats; ensuring RINEX 3.xx conformity.

But a lot of additional work load is generated by secondary (not so obvious) issues like adapting and controlling all output files, error and warning messages, and format statements due to new observation types, linear combinations, and an increased number of satellites, observations, and parameters; enabling an arbitrary number of GNSS in the programs (e.g., loops and control structures must be generalized); customizing the GUI to enable all new features and updating all corresponding help files; adapting output extraction programs to new formats.

Finally, all the technical issues must be considered, too. Work-intensive examples are switching from a static to a fully dynamic memory management; removing all hardwired dimension limits; consequently making use of modern programming features like function and operator overloading.

For the redesign of the Bernese GPS Software, about 50 from 103 programs, 800 from 1200 subroutines, 15 file formats, and most GUI panels and output files must be changed, updated, or rewritten.

Including the conceptual and design phase, the resulting effort is estimated to be about six man-years.

5. SUMMARY AND OUTLOOK

With several projected next-generation GNSS, updates of existing systems, and new innovations in satellite navigation, the need of scientific multi-GNSS analysis software packages arises.

A list of requirements for a modern GNSS analysis tool was composed and presented. Attention was turned not only to scientific but also to technical issues. Of course, the list of requirements is always subject to change and matures with the software development process and innovations in the field of GNSS data processing and observation modelling. Conceptual considerations of the software design followed, based on the list of requirements. The fundamental software architecture and several basic design principles were presented and discussed.

The update of the Bernese GPS Software to full multi-GNSS capability was used as an example of actual development work. Necessary steps and changes were addressed, drawing special attention to a lot of work-intensive secondary issues. The overall effort for the

update was estimated to be about six man-years. This new updated multi-GNSS version of the BSW will replace the current release in near future.

In summary, it was shown that the development of a multi-GNSS software for scientific purposes is a very demanding and costly task; but nevertheless a task being well worthwhile, opening up a variety of interesting scientific research fields.

6. REFERENCES

1. Dach, R., Brockmann, E., Schaer, S., Beutler, G., Meindl, M., Prange, L., Bock, H., Jäggi, A. & Ostini, L. (2009). GNSS processing at CODE: status report. *Journal of Geodesy* **83**(3-4), 353-366.

2. Dach, R., Hugentobler, U., Fridez, P. & Meindl, M. (Eds) (2007). *Bernese GPS Software Version 5.0*, Astronomical Institute, Universtiy of Bern, Bern, Switzerland.

3. Gurtner, W. & Estey, L. (2007). RINEX - The Receiver Independent Exchange Format Version 3.00 Online at *http://igscb.jpl.nasa.gov/igscb/data/format/rinex300.pdf* (as of 25. August 2009).

4. Ray, J., Altamimi, Z., Collilieux, X. & van Dam, T. (2008). Anomalous harmonics in the spectra of GPS position estimates. *GPS Solutions* **12**(1), 55-64

5. Habrich, H. (1999). *Geodetic Applications of the Global Navigation Satellite System (GLONASS) and of GPS/GLONASS Combinations*, Ph.D. dissertation, Astronomical Institute, University of Bern, Bern, Switzerland.