International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XXXIX-B1, 2012
XXII ISPRS Congress, 25 August – 01 September 2012, Melbourne, Australia

THE NEED AND KEYS FOR A NEW GENERATION NETWORK ADJUSTMENT SOFTWARE

I.Colomina[1], M.Blázquez[1], J.A.Navarro[1], J.Sastre[2]

[1]Institute of Geomatics, Generalitat de Catalunya & Universitat Politècnica de Catalunya
Av. Carl Friedrich Gauss 11, Parc Mediterrani de la Tecnologia, Castelldefels, Spain, ismael.colomina@ideg.es
[2]GeoNumerics, Josep Pla 82, Barcelona, Spain, jaume.sastre@geonumerics.com

KEY WORDS: network adjustment, orientation, calibration, modeling.

ABSTRACT:

Orientation and calibration of photogrammetric and remote sensing instruments is a fundamental capacity of current mapping systems and a fundamental research topic. Neither digital remote sensing acquisition systems nor direct orientation gear, like INS and GNSS technologies, made block adjustment obsolete. On the contrary, the continuous flow of new primary data acquisition systems has challenged the capacity of the legacy block adjustment systems —in general network adjustment systems— in many aspects: extensibility, genericity, portability, large data sets capacity, metadata support and many others. In this article, we concentrate on the extensibility and genericity challenges that current and future network systems shall face. For this purpose we propose a number of software design strategies with emphasis on rigorous abstract modeling that help in achieving simplicity, genericity and extensibility together with the protection of intellectual proper rights in a flexible manner. We illustrate our suggestions with the general design approach of GENA, the generic extensible network adjustment system of GeoNumerics.

1 INTRODUCTION

Orientation and calibration of photogrammetric and remote sensing instruments in their various modes, from direct to integrated, from geometric to radiometric, is a critical capacity of modern mapping systems and continues to be an active field of academic research. One of the key components of this capacity is the more than 200 year old method of network adjustment based in turn on the even older least-squares estimation method. Contrary to what the introduction of electronic light sensors made us fear or hope, digital cameras did not kill block adjustment for sensor calibration. Contrary to what the introduction of navigation technologies, mainly GPS and INS, made us fear or hope, INS/GPS did not kill block adjustment for sensor orientation. Furthermore, even in the strict navigation domain, there are environments and circumstances, where navigation cannot be performed only with "navigation instruments" and where the "imaging instruments" —in principle not designed for navigation purposes— have to come to the rescue. Thus, concepts like vision-aided navigation and Simultaneous Localization and Mapping (SLAM) have become fundamental in the field of robotics navigation (Leonard and Durrant-Whyte, 1991, Smith et al., 1986).

In the geomatic field, it is well known that the network adjustment method started with geodetic surveying for horizontal terrestrial networks at the beginning of the 19th century. In the second half of the 20th century it was applied to geodetic photogrammetry for global 3D networks (Schmid, 1974), to close-range photogrammetry (Brown, 1956, Brown, 1971), to aerial photogrammetry (strip, independent model and bundle adjustment), and reached remote sensing with the SPOT satellites (Kratky, 1989). Airborne laser scanning (Kager, 2004, Frieß, 2006, Skaloud and Lichti, 2006), terrestrial mobile mapping (Jansa et al., 2004), radiative transfer modeling and radiometric calibration (Chandelier and Martinoty, 2009, Honkavaara et al., 2009), and spatio-temporal orientation and calibration (Blázquez, 2008, Blázquez and Colomina, 2012b) are recent new beneficiaries from the method.

The network adjustment method has also stepped into the realm of INS/GPS post-processing. Indeed, until recently, the method was used for static problems; i.e., to estimate unknown parameters that are not time dependent (random variables) and that are related to observations by observation equations (stochastic equations). However, in 2004-2005 (Colomina and Blázquez, 2004, Colomina and Blázquez, 2005) we introduced the "dynamic network" concept where unknown parameters can be both time independent (random variables) and time dependent (stochastic processes), and where observations can be related to parameters by both static observation equations (stochastic equations) and by dynamic observation equations (stochastic differential equations). Classical [static] networks are a particular case of dynamic networks. The dynamic network (DN) method is an alternative to Kalman filtering and smoothing (KFS) for non real-time estimation tasks. DN is more flexible than KFS and supports models involving unknowns at different time epochs. The DN method has been proposed for airborne gravimetry (Térmens and Colomina, 2004) and for trajectory determination in terrestrial mobile mapping (Rouzaud and Skaloud, 2011).

That DN and KFS are two different ways to solve the same problem indicates that the network adjustment method is also a modeling technique, where problems are formulated in terms of four fundamental entities: instruments, observations, parameters (random variables and stochastic processes) and models (stochastic equations). Consequently, we will talk about "network modeling and adjustment" rather than network adjustment and will shortly refer to it as the "network approach" or "network method." Analogously we will talk of "network software" when referring to network modeling and adjustment software.

From a software point of view, the long and successful history of the network approach is that of one of changing requirements. In geomatics and everywhere else, changing requirements, beyond certain point and in the absence of software development methods, can make clean software designs become a tangle. It is understandable then, that the big success of the network method has translated into the big challenge of the network software. Unfortunately, though understandably and not always (Colomina et al., 1992, Elassal, 1983), most network software packages are designed, developed and commercialized for niche geomatic markets (geodetic surveying, surveying, aerial photogrammetry, close-

range photogrammetry, satellite-based remote sensing, aerial laser scanning, terrestrial laser scanning, terrestrial mobile mapping systems, etc.). This fragmentation makes software development and evolution unnecessarily expensive, error-prone and does not let different geomatic measurement techniques be easily integrated. Moreover, with today's wave of new sensing techniques and observation platforms, refactoring[1] and/or patching the software is too long and expensive an option every time that a new instrument or configuration appear.

In general, changing of requirements together with old, legacy, or poorly designed code make the software to rot. Because of this (Martin, 2002), some software designs are said to suffer from the seven deadly smells, namely: rigidity, fragility, immobility, viscosity, needless complexity, needless repetition and opacity. *Rigidity* means that changing one part of the system forces changes in other parts; *fragility* that changes in one part cause bugs in unrelated parts; *immobility* that system components cannot easily be isolated for reuse; *viscosity* that doing things right is harder than doing things wrong; *needless complexity* that there is infrastructure or abstraction without immediate, direct benefit; *needless repetition* that there is repeated code that could be unified under a single abstraction; and *opacity* that the code is hard to read or understand.

These problems affect most software systems and that network software aged in the past is understandable. Today, however, software engineering is a mature discipline and the drifting of the requirements cannot longer be blamed for the degradation of the design. The capability to cope with changing user requirements is a system design requirement of modern software systems which translates into one of the fundamental challenges of software design: achieving simplicity and extensibility.

The above considerations make us believe that modern network software shall be generic and extensible. In software engineering, the algorithms of *generic* software are written in terms of abstract types that are instantiated as required for specific types provided as parameters. *Extensible* software is software prepared for future growth.

Last not least, there is the design challenge of promoting the use and growth of software while protecting the intellectual property. It is frequently the case that new instrument models cannot be included into the established network software: the one who knows the instrument and its model does not have access to either source code or extensible software and may not be interested in providing the models for integration into existing network software. The one who owns the software may not be interested in extending it if sales are not guaranteed, particularly if it is not extensible. Beyond generic and extensible, network software shall be designed in a way that balances fast circulation and growth, fair compensation of everyone's effort and protection of knowledge and intellectual property.

This paper discusses how the previously identified needs of network software

- simplicity,
- genericity,
- extensibility, and
- protection of intellectual property

[1] Refactoring is the disciplined practice of changing the structure of code without changing what it does.

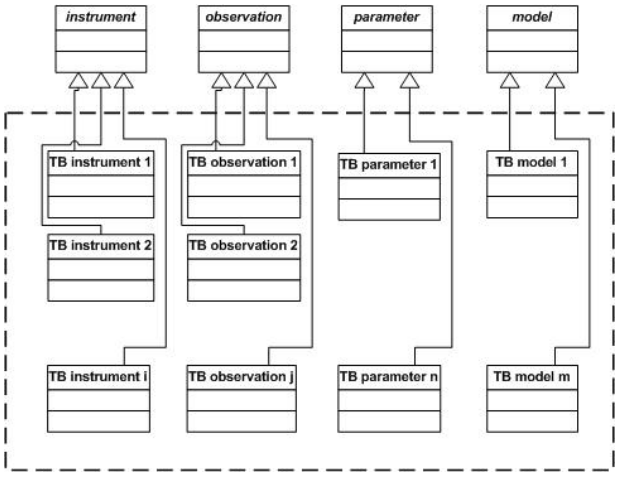

Figure 1: The GENA reference model and inheritance scheme.

can be achieved and illustrates the process by referring to GeoNumeric's network software GENA. GENA stands for Generic Extensible Network Approach, is a product of GeoNumerics and the Institute of Geomatics has developed a significant part of it under contract.

This paper is organized as follows. Sections 2 and 3 discuss software simplicity, genericity and extensibility, section 4 is devoted to generic and extensible network software architectures, section 5 provides references to use cases where the same software system (GENA) was used for different network types and section 6 summarizes the article ideas.

2 BALANCING SIMPLICITY AND EXTENSIBILITY

Software simplicity and extensibility have to be balanced. For a correct, consistent and complete software system, simplicity is obviously harder to achieve than complexity. Simple and extensible design requires correct domain models and well tuned abstraction levels. Correct domain modeling requires technical specialization as well as context awareness. Abstraction requires an additional effort to find common traits and behaviors of domain entities. Insufficient or needless abstraction leads to complex systems. Wrong abstraction leads to non extensible systems. Correct abstract models are, therefore, the key to simple and extensible systems.

In GENA, the main abstract model is the fundamental reference model based on four abstract classes: instruments, observations, parameters and mathematical models (figure 1). In other words, the fundamental data types of GENA are grouped in four categories: instruments, observations, parameters and mathematical model types. A concrete type, like for instance "GNSS receiver" will inherit from the abstract data type "instrument." A new instrument would probably generate measurements (observations) different from others. These measurements would be probably related to some unknown parameters of interest like points or orientation parameters and to some other unknown parameters like the self-calibration ones. The relation between the observations and the parameters, possibly with the participation of the instrument constants, is materialized by the mathematical models (stochastic equations).

"All" what has to be done to deal with a new instrument or system of instruments is to identify the instrument and its characteristic values, to characterize the type of its related measurements, to

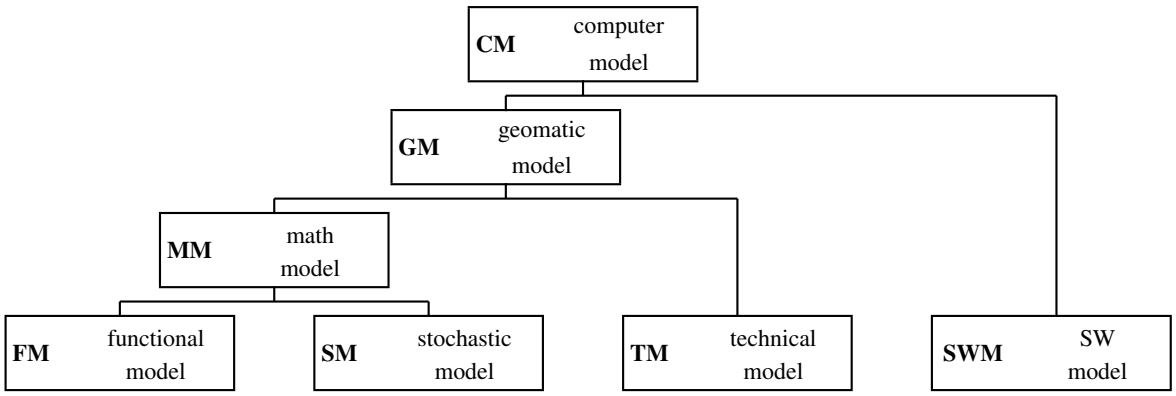

Figure 2: A modeling hierarchy in a network software system.

identify the parameters and the mathematical models that relate them. Once this is done, specific classes that inherit from the fundamental reference model can be reused or newly developed (figure 1). In other words, extending GENA to cope with new types of instruments, measures or mathematical models reduces to creating new data types. This is "all."

In practice, however, there are other aspects that count beyond the mathematical models and the other fundamental modeling entities (instruments, observations and parameters). In GENA, these "other aspects" correspond to other abstraction levels and models as depicted in figure 2. Thus, easy extensibility of input and output components, extensibility of metadata specification, support for physical units in the input and output (IO) tasks and internal computations and standardization of analysis tools, from outlier detection to numerical convergence criteria have to be tackled. In this case, it is "simplicity" for the developer of extensions that prevails. The approach taken by GENA is that, once the modeling entities are built, its software framework shall provide resources so that the above and other possible tasks are solved in a transparent (automatically, not known to the developer and user) or generic (activated by the data types or data type codes) way.

3 BALANCING GENERICITY AND EXTENSIBILITY

Balancing genericity and extensibility is deciding and designing what shall be newly coded —an extension action— and what shall work automatically upon reception of a data type argument —a generic behavior.

Extensibility and genericity are mutually dependent. In object-oriented design, for instance, the effective construction of a new data type requires the implementation of its interface. Thus, if the system design is based on correct abstract models, the standardized interface of the abstract data types guarantees that specific data types provide the necessary and sufficient information for other software components to behave generically.

In GENA, the four fundamental abstract data types provide the interface that allows the IO, least-squares estimation, numerical control, quality control, etc. components to behave generically. The specific, instantiated data types are grouped in "model toolboxes." The set of all generic components are grouped in the GENA software platform that does not change when new instruments/measurements have to be modeled and the models have to be coded. Thus, in GENA, the balance between extensibility and genericity is achieved by the "GENA model toolboxes" and by the "GENA platform." The model toolboxes, or more to the point the four abstract data types, their interface and the inheritance mechanisms, contribute the extensibility capacity. The platform provides the genericity capacity.

4 ON THE SOFTWARE FRAMEWORK AND ARCHITECTURE FOR GENERICITY AND EXTENSIBILITY

A software framework is a set of software libraries that expose a well defined application programming interface (API). The goal of a software framework is to facilitate the development of applications, products and solutions by isolating the domain developers from low-level programming details. In other words, a software framework is an abstraction in which software providing *specific* —and in the application domain context *generic*— functionality can be used for the medium- and high-level development of domain-specific applications.

The software architecture of a system is an abstraction that describes the software components, the relations among them and the properties of both components and relations.

In order to describe our vision of a modern network software, we further describe the modeling concepts of figure 2 through the following definitions.

Mathematical Model (MM): is an abstract model that uses mathematical language to describe the behaviour of a system. In the context of this article, a mathematical model is a functional model plus and stochastic model.

Geomatic Model (GM): is an extension of a mathematical model that includes coordinate reference frames, units of measurements, thresholds and any other information required to describe the geomatic properties of the mathematical model entities.

Technical Model (TM): In our context, a technical model is the difference between a geomatic model and a mathematical model.

Software Model (SWM): is an abstract model that describes a computer programme or software system, usually with a special graphical modeling language.

Computer Model (CM): is a computer program that simulates an abstract model of a particular system. (In a network adjustment system, for instance, the network computer model simulates the [abstract] network geomatic model.)

With the above definitions, we can now define the *Network Model* as a computer model of a physical reality; i.e., a computer model

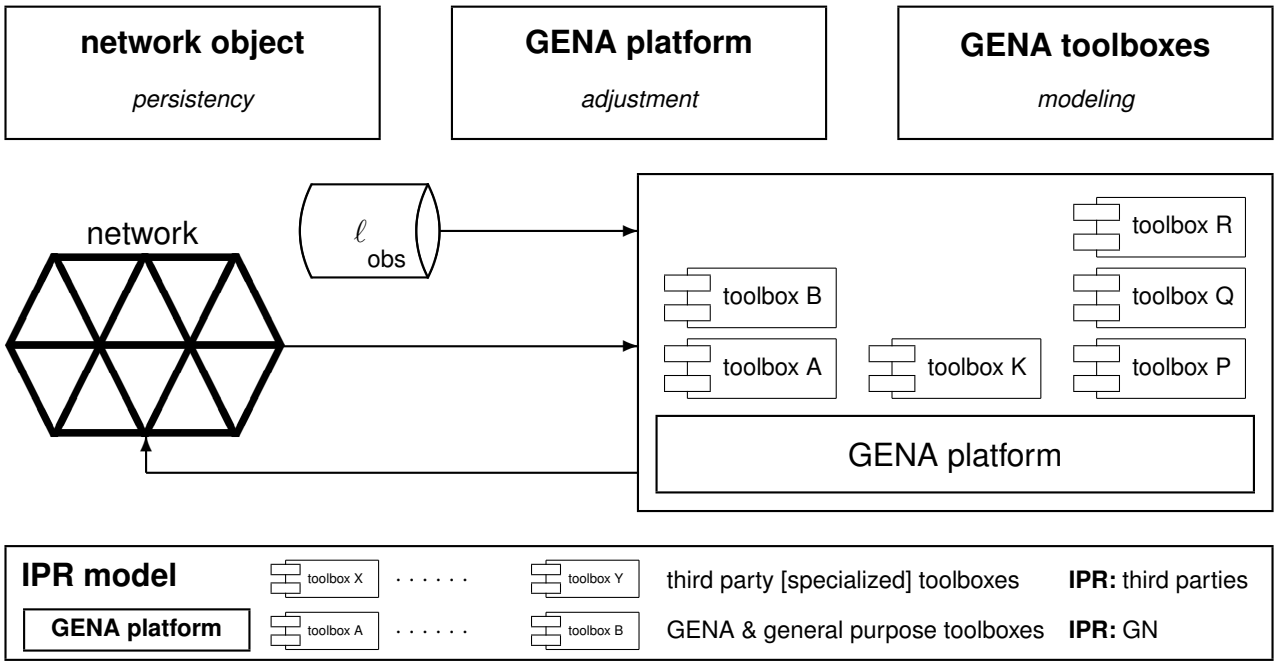

Figure 3: Network-centric architecture and IPR model.

of a of a set of measurements, the involved instruments and unknown parameters, and the mathematical models that relate them. (Note that the network model is a function of each particular network.)

Thus, in GENA, the mathematical models are materialized in the *GENA model toolboxes*, the *generic GENA software platform* is built with libraries of the *GENA software framework* that materialize the technical model and the software model corresponds to the software structure which includes the "network" data type.

The network data type models a network object that is both transient —exists as an entity during a network adjustment— and persistent —remains as a file after a network adjustment. The concept of persistent network objects or network files is instrumental in designing network-centric architectures as opposed to traditional process based ones (figure 3). In a network-centric architecture, input data are set into the network, the network is adjusted and, later on, the adjusted data can be extracted and adjustment reports can be generated.

4.1 Generic and dialectal man-machine interfaces

Most software systems communicate to other software systems and to human beings. To cover these two interface actors, the following types of interfaces are usually employed:

- Application Programming Interface (API);
- File Interface (FI);
- Command Line Interface (CLI); and
- Graphical User Interface (GUI).

In general, the API is used to communicate to other software systems, the GUI to humans, and the FI and CLI to both.

We will not devote a special attention to the API, as it can be seen as one more component of the generic software framework. We will also not discuss the CLI. A CLI uses to accept a number of arguments and behave generically with respect to them. The GUI and FI are of interest because

(a) the GUI interacts with a human operator who cannot be asked to behave generically, and

(b) ideally, the extension of the network software through the creation of new data types should

  (b.1) not require the coding of new IO software and
  (b.2) allow for the use of existing standards or de facto standards with a reasonable coding effort.

In the GUI case, if we assume that the FI can be generic, particularly for production environments where productivity and specialization are high, a reasonable approach is that

(a.1) the GUI is built around the generic FI (the GUI speaks the language of the generic platform when communicating "inwards" to the platform), and

(a.2) the GUI adapts to the particular domain environment (the GUI speaks the language of the domain —surveying, photogrammetry, remote sensing, etc.— when communicating "outwards" to the user),

which translates into building specific GUIs if the size of the targeted user segment and market permits.

In the case of the FI, for the input of measurements and the output of adjusted measurements, residuals and parameters, it is possible to specify formats parameterized by the network data types —therefore generic formats— that can be used in the IO operations. These formats can be generic or *open* because of their ability to represent any kind of network data type, which paves the way to automated, uniform IO.

The generic format specification can always be the same no matter the network data to read or write. However, the specific format to read or write depends on the particular data type. In the case of GENA, its generic software platform —also taking care of IO operations— may retrieve the particular characteristics of each network data type interrogating these in execution time. This makes the formats *closed* and fully operational in spite of their openness (genericity and extensibility). Moreover, in GENA,

the format specification for instruments, observations, parameters and [formal input of] model data are the same. In this way, for instance, the adjusted calibration parameters output of one adjustment can be read as input instrument calibration constants in the next one.

The addition of new network data types implies no change in generic network platforms. The characterization of new network model data types as described in section 2 is enough to integrate these in the FI. This contributes to simplicity and mitigates the burden of the unavoidable development of software bridges to legacy and/or external formats.

4.2 Generic metadata

Bridging the gap between genericity and a truly operational FI goes beyond giving the generic platform the ability to delegate to the network data types the responsibility of providing with the specific details related to IO. Lack of data metadata is behind many delays and data processing accidents.

The FI must also make room for metadata describing key aspects of the data included. Some of these aspects will be structural ones while other will describe variable traits of the information.

Structural metadata must describe aspects so important as the identification of the network data types included in the files —so the built-in, automated IO mechanisms may be triggered correctly and automatically. Variable metadata takes care of no less relevant aspects that, however, may not be considered as structural traits of data, as, for instance, the physical units used in a particular measurements data set, or the coordinate reference frames in use, or the role played by a parameter —constant or variable— in a particular adjustment.

4.3 Generic numerical kernel

Scientific computing can also benefit from modern computer science and software engineering (Dubois, 1997). And, although modern programming languages generate executable code that is no faster than that generated by old FORTRAN and C, the use of their programming mechanisms and available libraries often result in higher performance. That said, the numerical kernel of a network software platform has to implement the functions of a robust non-linear least-squares (RNLLS) estimator for large data sets of observations and parameters —i.e., threshold-monitored iteration of the cycle linearization, solution (for parameters and residuals), detection and removal of outliers, covariance estimation, variance component estimation, etc.— in one of its many possible forms.

Once the numerical- and statistical-control threshold abstractions are defined to let the generic convergence- and statistical-monitoring algorithms work, the construction of a generic RNLLS estimator is a common practice in numerical analysis and programming. There are many examples thereof like the generic subroutines, functions or methods to find polynomial roots, solve differential equations or optimize functions. In the case of GENA, we follow the standard generic numerical approach where the mathematical functional model g, where $g(\ell + v, x_1, \ldots, x_n) = 0$, is passed as a function argument with standardized interfaces to retrieve, given observations ℓ and parameters $x_1, \ldots, x_n$, the value of g and its jacobian matrix with respect to $\ell, x_1, \ldots, x_n$. Also, in the case of GENA, for the sake of modeling simplicity and flexibility, the Gauß-Helmert formulation ($g(\ell + v, x_1, \ldots, x_n) = 0$) is preferred over the more restrictive Gauß-Markov one ($\ell + v = f(x_1, \ldots, x_n)$) and a mechanism to numerically compute jacobian matrices is provided.

In general, there is the open issue of parameter initial approximations' generation for the initialization of the iterative RNLLS solver. Most non-trivial mathematical models in geomatics are non-linear. Generally speaking, there is no universal method to compute initial approximations for non-linear problems, even not to network adjustment problems restricted to geomatics. Thus, in the best of cases, a generic RNLLS initializer may be provided as part of the generic network platform with applicability limited to a subset of models and circumstances.

4.4 Generic numerical and statistical control

Numerical and statistical control for various purposes is an essential feature of a RNLLS solver. Making numerical and statistical decisions, like declaring numerical convergence or divergence, or accepting or rejecting a measurement, is not a minor issue for disparate data sets and network geometries. Therefore, the abstract data types shall be able to provide this information to the generic algorithms. In most cases this design aspect corresponds to the technical model (TM, see figure 2) as answering questions like "how small is small?" depends on technical domain aspects; for instance, a correction to an Earth reference frame transformation parameter can be considered small below few millimeters whereas a correction to a camera principal point may only be acceptable below few micrometers. These, and other similar abstractions for the TM are often neglected in spite of their practical relevance.

5 USE CASES

The generic and extensible approach discussed throughout the paper has been demonstrated in practice by using GENA for various orientation and calibration tasks: simultaneous orientation and calibration of frame camera and laser scanners (Angelats et al., 2012), radiative transfer modeling and camera radiometric calibration (Antequera et al., 2012), new camera orientation methods like the use of relative aerial control (Blázquez and Colomina, 2012c), spatio-temporal calibration of frame cameras (Blázquez and Colomina, 2012b), or quasi-direct orientation (Fast AT) (Blázquez and Colomina, 2012a). Also, the similar and early approach reported in (Colomina et al., 1992) has been successfully applied to geodetic surveying, airborne photogrammetry, satellite remote sensing and to airborne gravimetry simulations with the dynamic network method (Térmens and Colomina, 2004).

6 SUMMARY AND CONCLUSIONS

The construction of modern network adjustment software able to cope with today's continuous flow of new geomatic instruments is a typical software engineering task of dealing with changing requirements where simplicity, genericity, extensibility and performance have to be balanced. In the article, we have tried to demonstrate that this software engineering task is, above all, a modeling exercise where the identification of a network's abstract data types is the main part. We have illustrated the preceding statement with the GENA fundamental reference model that is based on four abstract data types: instruments, observations, parameters and models. If a complete and correct collection of abstract data types is provided, it is possible to construct software that is both generic and extensible. We have illustrated this with a software architecture concept that separates the common network adjustment functions from the particularities of each instrument and its associated measurements and models. The concept concentrates the common functions in a *generic network adjustment platform* and the instruments' particularities in *model toolboxes*

that, in the GENA case, include the specific instrument, observation, parameter and model data types.

The proposed concept, as indicated, has been implemented in GeoNumeric's GENA software platform and used, with the corresponding specific model toolboxes, for different network adjustment use cases including radiative transfer modeling and radiometric calibration, spatio-temporal orientation and calibration of cameras, orientation and calibration of airborne laser scanners, and combined orientation and calibration of airborne laser scanners and cameras.

REFERENCES

Angelats, E., Blázquez, M. and Colomina, I., 2012. Simultaneous orientation and calibration of images and laser point clouds with straight segments. In: International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Melbourne, Australia.

Antequera, R., Andrinal, P., Colomina, I., Navarro, J. and Pros, A., 2012. Corrección radiométrica automática de imágenes generadas por cámaras matriciales ópticas. In: X Congreso TOPCART 2012: Congreso Iberoamericano de Geomática y Ciencias de la Tierra, Madrid, Spain.

Blázquez, M., 2008. A new approach to spatio-temporal calibration of multi-sensor systems. In: International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. 37-B1, Beijing, China.

Blázquez, M. and Colomina, I., 2012a. Fast AT: a simple procedure for quasi direct orientation. ISPRS Journal of Photogrammetry and Remote Sensing.

Blázquez, M. and Colomina, I., 2012b. On INS/GNSS-based time synchronization in photogrammetric and remote sensing multi-sensor systems. PFG Photogrammetrie, Fernerkundung und Geoinformation 2012(2), pp. 91–104.

Blázquez, M. and Colomina, I., 2012c. Relative INS/GNSS aerial control in integrated sensor orientation: models and performance. ISPRS Journal of Photogrammetry and Remote Sensing 67(1), pp. 120–133.

Brown, D., 1956. The simultaneous determination of the orientation and lens distortion of a photogrammetric camera. Air Force Missile Test Center Report 56-20, Patrick AFB, Florida, USA.

Brown, D., 1971. Close range camera calibration. Photogrammetric Engineering 37(8), pp. 855–866.

Chandelier, L. and Martinoty, G., 2009. Radiometric aerial triangulation for the equalization of digital aerial images and orthoimages. Photogrammetric Engineering and Remote Sensing 75(2), pp. 193–200.

Colomina, I. and Blázquez, M., 2004. On the stochastic modeling and solution of time dependent networks. In: Proceedings of the VI International Geomatic Week, Barcelona, Spain.

Colomina, I. and Blázquez, M., 2005. On the stochastic modeling and solution of time dependent networks. In: Proceedings of the VI International Geomatic Week, Barcelona, Spain.

Colomina, I., Navarro, J. and Térmens, A., 1992. GeoTeX: a general point determination system. In: International Archives of Photogrammetry and Remote Sensing, Vol. 29-B3, International Society for Photogrammetry and Remote Sensing, pp. 656–664.

Dubois, P., 1997. Object technology for scientific computing. The object-oriented series, Prentice Hall, Upper Saddle River, NJ, USA.

Elassal, A., 1983. Generalized adjustment by least squares (GALS). Photogrammetric Engineering and Remote Sensing 49, pp. 201–206.

Frieß, P., 2006. Toward a rigorous methodology for airborne laser mapping. In: Proceedings of the EuroCOW 2006, European Spatial Data Research - EuroSDR, Castelldefels, Spain.

Honkavaara, E., Arbiol, R., Markelin, L., Martínez, L., Cramer, M., Bovet, S., Chandelier, L., Ilves, R., Klonus, S., Marshal, P., Schlpfer, D., Tabor, M., Thom, C. and Veje, N., 2009. Digital airborne photogrammetry –a new tool for quantitative remote sensing?– a state-of-the-art review on radiometric aspects of digital photogrammetric images. Remote Sensing 1, pp. 577–605.

Jansa, J., Studnicka, N., Forkert, G. and Haring, A. Kager, H., 2004. Terrestrial laserscanning and photogrammetry — acquisition techniques complementing one another. In: International Archives of Photogrammetry and Remote Sensing, Vol. 35-B7, International Society for Photogrammetry and Remote Sensing, pp. 948–953.

Kager, H., 2004. Discrepancies between overlapping laser scanning strips - simultaneous fitting of aerial laser scanner strips. In: International Archives of Photogrammetry and Remote Sensing, Vol. 35-B1, International Society for Photogrammetry and Remote Sensing, pp. 555–560.

Kratky, V., 1989. Rigorous photogrammetric processing of SPOT images at CCM Canada. ISPRS Journal of Photogrammetry and Remote Sensing 53(9), pp. 1223–1230.

Leonard, J. and Durrant-Whyte, H., 1991. Simultaneous map building and localization for an autonomous mobile robot. In: Proceedings of the International Workshop IROS'91, Vol. 3, IEEE/RSJ, pp. 1442–1447.

Martin, R., 2002. Agile software development: principles, patterns, and practices. Prentice Hall, Hemel Hempstead, UK.

Rouzaud, J. and Skaloud, J., 2011. Rigorous integration of inertial navigation with optical sensors by dynamic networks. Navigation 58(2), pp. 141–152.

Schmid, H., 1974. Worldwide geocentric satellite triangulation. Journal of Geophysical Research 79(35), pp. 5349–5376.

Skaloud, J. and Lichti, D., 2006. Rigorous approach to boresight self-calibration in airborne laser scanning. ISPRS Journal of Photogrammetry and Remote Sensing 61, pp. 47–59.

Smith, R., Self, M. and Cheeseman, P., 1986. Estimating uncertain spatial relationships in robotics. In: Proceedings of the Second Annual Conference on Uncertainty in Artificial Intelligence. UAI '86, University of Pennsylvania, Philadelphia, PA, USA, pp. 435–461.

Térmens, A. and Colomina, I., 2004. Network approach versus state-space approach for strapdown inertial kinematic gravimetry. In: Gravity, Geoid and Space Missions - GGSM2004, Porto, Portugal.