A Generic, Extensible Data Model for Trajectory Determination Systems

José A. Navarro[1], M. Eulàlia Parés[1], Ismael Colomina[2] and Marta Blázquez[2]
[1]*Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), Av. Carl Friedrich Gauss, 7. Building B4, 08860 Castelldefels, Spain*
[2]*GeoNumerics, S. L., Av. Carl Friedrich Gauss, 11. Building B6, 08860 Castelldefels, Spain*

Keywords: Data Model, Generic, Extensible, Trajectory Determination System.

Abstract: Trajectory Determination Systems (TDS) use the measurements delivered by sensors to estimate trajectories. Many are the types of sensors that may be used already by such systems and new ones appear constantly in the market. Variety and fast change pace pose a problem for the maintenance of any kind of software system, including TDSs. Some data models and / or standards dealing with sensor data exist, but these are either too generic (and ambitious) or, on the contrary, targeted at very specific types of observations (as, for instance, GNSS.) This paper introduces a compact but complete, generic and extensible data model powerful enough as to deal with the kind of observations involved in trajectory determination and able to alleviate or even eliminate the software maintenance toll derived by constant changes in data sources. Two materialization of this data model, its file and network interfaces are briefly presented here as well as a portable C++ library implementing such interfaces.

1 INTRODUCTION

Trajectory Determination Systems (TDS) (Titterton and Weston, 2004), at least in the context of this paper, are software applications that compute trajectories using the measurements provided by one or, more usually, several sensors, being a *trajectory* the path of a moving object. TDSs are becoming more and more used in several areas, as for instance precision agriculture (Karpowicz, 2016; GSA, 2015), unmanned aircraft navigation (Colomina and Molina, 2014), indoor positioning (Zhu et al., 2014) or even autonomous vehicle driving (Minh, 2014), among others. Normally, these applications need not a single sensor but a set of these to deliver satisfactory results because of the degree of complexity that some navigation scenarios may reach.

At the same time, sensors, the data sources used by TDSs, evolve very quickly (Groves et al., 2014a; Groves et al., 2014b). New ones appear on the market, others are improved. TDSs must adapt quickly too to these changes to maximize their potential to derive quality trajectories; this is especially true in very competitive markets. These new or improved sensors must be modelled and integrated into the TDS minimizing delays as much as possible. This poses a problem showing different facets: scientific, software engineering and interfaces. Although this applies to any software system in general, one possible solution is to develop TDSs that may be extended easily, as for instance NAVEGA (Parés and Colomina, 2015). In this TDS, the internal estimation engine or external data interface have been designed in such a way that including new models or data types implies no changes in the software. RIFE (Soloviev and Yang, 2013) uses a similar approach. Another example of a generic TDS is the FLY and SPIDER suite (Marietta et al., 2015).

Extensibility has long been the target of several contributions in Geomatics (Tscherning, 1978; Elassal, 1983; Sarjakoski, 1988; Crippa et al., 1989; Colomina, 1992). However, it is surprising to see how little the problem of generic data interfaces has been treated. The works on ISO standards in this area (Kresse and Fadaie, 2004) are loosely related to the problem tackled here. Very recently, though, the UAID generic sensor interface for plug-and-play navigation systems (Soloviev et al., 2016) refers to the Defense Advanced Research Project Agency's (DARPA) "ASPN Interface Control Documents" whose aim is the development of techniques to integrate low cost sensors under the umbrella of a plug & play architecture. DARPA's interest in this topic should confirm its relevance.

There exist several standards covering sensor and observation data models as those from the Open GIS

Navarro, J., Parés, M., Colomina, I. and Blázquez, M.
A Generic, Extensible Data Model for Trajectory Determination Systems.
DOI: 10.5220/0006258400170025
In *Proceedings of the 3rd International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM 2017)*, pages 17-25
ISBN: 978-989-758-252-3

Consortium (OGC, 2007; OGC, 2013). The scope, however, of these is too wide, their proposal too complex to achieve efficiently the goals of this work. Data or metadata files written according to these recommendations would be far too complicated for the regular TDS user. On the other side, standards covering data for *specific* sensors also exist, as the RINEX format (and its relatives, ANTEX, IONEX and SP3) for GNSS measurements (IGS, 2013) or the LAS format for point clouds derived from laser scanning (ASPRS, 2009). But since these formats are targeted at specific sensors, it is not possible to use them as the base of a generic data model.

The aim of this paper is to introduce ASTROLABE, a generic, extensible and complete but concise data model specifically targeted at TDS data and metadata, able to cope with change and innovation. The development of ASTROLABE started in 2009 at the former Institute of Geomatics and, since 2014, it has been continued at the Geomatics Division of CTTC with the cooperation of GeoNumerics. There exist two *translations* of this data model that are materialized as a file and a network interfaces as well as a portable library written in C++ implementing these. Further sections of this work will describe these components.

2 THE ASTROLABE DATA MODEL

ASTROLABE means to be complete. In this context, this means that all the aspects related to data must be taken into account so no assumptions about their properties need to be done. We are talking about metadata, a very important facet that, unfortunately, is often neglected. ASTROLABE takes care of defining both data and related metadata. Examples of metadata might be the units used in measurements or the coordinate reference systems these are referred to. The following sections will describe (1) how data is modelled to achieve genericity and extensibility and (2) how metadata is specified to fulfill the completeness goal.

2.1 About Data

The determination of a trajectory, as any other process, takes some *inputs*, performs some kind of *process* and then delivers some *outputs*. (Previous) states, observations and instrument data are the inputs in the case of ASTROLABE. Models represent the process while (new) states are the outputs. More formally these are the pillars on which the data model relies:

- Instrument (a.k.a. sensor.) Device used to measure some observable, thus delivering observations. ASTROLABE defines an instruments as the set of constants that characterizes it. A GNSS receiver is a typical example of instrument.
- Observable and observation (a.k.a. measurement.) An observation or measurement is one of the values that an observable may take. An observable is a numerical property of a physical system that may be measured. Technically, it is a random variable. An example of observable in this context is a linear acceleration; the corresponding observation or measurement could be 12 m/s^2.
- (Mathematical) model. Mathematical description of a process. The logic that transforms previous states and observations into new states (see below.) Mathematical models are not data by themselves, but are included here for the sake of completeness and coherence in the description of the data model. Model identifiers, however, are used in observation equations (see below) to relate observations, instruments and models themselves to describe how to estimate states.
- States. States are also random variables that have to be estimated from known (former) states and observations, instruments and mathematical models. Typical states in the context of TDSs are position or attitude random vectors.
- Observation equations. These relate observations, instruments, models and states. Observation equations describe how a new state is obtained, specifying both the necessary inputs (former states, observations and instruments) and the necessary procedures (the mathematical model.)

From the structural standpoint, observations and states are identical in ASTROLABE. The obvious difference is that while observations contain measurements, states hold the result of a computation. Therefore, the description of observations below is valid for these two entities.

These are the components integrating an observation (in fact, *any* kind of observation in ASTROLABE:)

observation: (event tag, active flag, identifier, instance identifier, time stamp, tags, expectations, covariance matrix)

The *event tag* is a marker to tell apart the different data entities making the ASTROLABE data model, as observations or observation equations; the *active flag* is used to activate (or deactivate) the observation so it is taken or not into account when computing states (this is the way to avoid the actual removal of the observation from a data file, for instance, and still be able to ignore it.) The *identifier* is an unique code stating the type of the observation (is this an IMU model *x* or an odometer model *y*?) The *instance identifier* is used to tell apart different instances of observations of the same type—sharing the same identifier; for instance, if two or more *identical* sensors (e.g., odometer model *w*) are used in an observation campaign, the observations coming from these will share the same identifier (since all of them come from the same kind of odometer) but will have different instance identifiers (to tell apart the different odometers.) The *time stamp* contains the time when the observation was obtained. The *tags* are a set of values that, despite not being an intrinsic part of the observation, help to complement it. For instance, some barometers may be affected by temperature; collecting the value of this observable at the moment the pressure was measured will improve the estimation of the output state. Obviously, barometers do not measure temperature; this is the reason why such magnitude plays the role of a tag in this observation. Note that tags are optional. Finally the values of the observation themselves (the *expectations*) and their *covariance matrix* complete the definition of the observation.

Providing that the error distribution of (whatever the) observations adheres to a Gaussian distribution, it will be possible to model them using the above structure. To be precise, Gaussian error distributions are a requisite set by CTTC's TDS, NAVEGA. ASTROLABE, in fact, supports a broader range of error probability distributions including all those for which the first and second moments exist.

The following is an example of observation:

```
l imu1 1 124.88 0.01 0.02 0.015 0.32 0.43 9.95
                1e-3 1e-3  1e-3 1e-2 1e-2 1e-2
```

The "l" marker denotes an observation; `imu1` is the identifier of the observation, pointing to some kind of inertial measuring unit; `1` is the instance identifier for this observation; if there were data coming for more (identical kind of) IMUs in the dataset, this instance identifier would tell these apart. The value `124.88` is the time tag. There are no tags. The remaining values in the first line above stand for the linear accelerations and angular velocities typically measured by IMUs. All the values in the second line stand for the covariance matrix of the observation (in this case, only standard deviations, no correlations.) Note that the active flag is not shown in the example.

In the next example, the following could be two valid observation corresponding to two identical temperature compensated barometers used simultaneously:

```
l tc_baro 1 124.88 25.4 1020.003 1.3
l tc_baro 2 124.88 25.3 1020.532 1.3
```

Focusing on the first of these two observations, the identifier `tc_baro` indicates the new kind of observation. Again, `1` is the instance identifier; the time stamp is again `124.88`. Note that this observation has a single tag, a temperature value, which for this case is `25.4`. The actual pressure value is `1020.003` and its standard deviation is `1.3`. The values for the second barometer (second observation) differ slightly from those provided for the first one. The important thing, however, is that the instance identifier for the second barometer is `2` and not `1`, so it is possible to tell apart the observations originating in different sensors.

To finish, the following could be an example of a position observation (longitude, latitude, height) retrieved from a GNSS receiver. No explanations (besides the fact that there are *no* tags in the observation) are given; all the components of the observation should be now clear to the reader.

```
l gnss_b 3 124.88 2.0003 41.0008 32.4
                    3e-2    3e-2 2e-1
```

Note how easily three completely different kinds of observations have been represented using the ASTROLABE data model.

As stated above, states adhere to the same structure just described, so this issue will be no further discussed. Instruments are defined using the same structure too. The main difference in this case is that instruments are considered as constants and their values, therefore, immutable. Their covariance matrices are interpreted as a mere indication of the quality of the constants defining the instruments.

The structure of an observation equation is the following:

observation equation: (event tag, activation flag, time stamp, model identifier, list of state instance identifiers, list of observation instance identifiers, list of instrument instance identifiers)

The meanings of the event tag, activation flag and time stamp fields have already been defined above. However, the event marker for observation equations is a lowercase letter "o" (instead of "l".) The model identifier points to the actual mathematical method to use to estimate a state out of the previous states, observations and instruments involved in the equation. Note the use of *instance identifiers* instead of mere identifiers. Doing so it is possible to refer to observations that, despite being of the same type, come from different actual sensors. The following could be an example of observation equation involving the `imu1` and `gnss_b` observations seen in the examples above:

```
o 124.88 compute_position 101 1 3
```

Again, the active flag has been omitted. `o` is the event marker and `compute_position` corresponds to the name of the model that will perform the transition from observations to states. The state that will be computed is the one pointed by the instance identifier `101` and to do it, the model will use the observations whose instance identifiers are, respectively, `1` and `3`. There are no instruments constants involved in this observation equation (these are optional.)

Observation equations may be seen as triggers, or the call to methods / routines specifying the set of input parameters needed to derive the output ones.

Finally, data is organized in different datasets (either disk files or network data streams.) Observations and observation equations go together in the observations data set; instruments constitute a separate one; output states are stored in their own dataset too.

2.2 About Metadata

The different examples shown in section 2.1 are clearly incomplete. The values related to time stamps, tags, expectations or their covariance matrix are not well defined. For instance, there is no information about the units used to express these magnitudes. Still further, there is no hint about reference frames or coordinate systems. Consequently, for instance, it is impossible to know if the temperature tag that accompanies the pressure reading delivered by the compensated barometer in the examples above is expressed in degrees Celsius or Fahrenheit.

ASTROLABE provides with metadata for all the data entities described in section 2.1—but observation equations, since these merely relate data entities. Additionally, metadata for time values (necessary for time stamps) are also provided. Metadata are always stored in XML files.

Below, the description of the several fields included in metadata are presented, separated by the different kinds of data entities found in the data model. Note that mathematical models are also characterized.

Observations and States. Metadata for these two data entities are identical. These are the items included:

Identifier. Unique code used to tell apart different kinds of observations or states. By means of this code, it is possible to relate actual observation or state data (which include the identifier) to the metadata that characterize them.

Dimension. Number of elements in the observation or states expectations vector.

Referencing. Code (or pairs of codes) identifying either the coordinate reference frame or reference frame plus coordinate system to which data is referred to.

Units. Used to state the units of each of the elements in the expectations vector (for observations or states.)

Default Covariance Matrix. Default covariance matrix for the expectations vector (observations or states.) Used when an observation (or state) includes no explicit covariance matrix values. This happens when a sensor does not provide covariance information and a default nominal value (usually stated by the manufacturer) must be used instead. The units are those of the expectations vector (for the standard deviations.) This is an optional metadata field.

Scale Factors. This optional field contains a list of positive scale factors for the standard deviations. When not present, all scale factor values default to 1.

Tag Metadata. This field is optional, as tags are. When present, the following items must be described:

- Number of tags.
- For each of these tags, the units and the referencing information (see above) must be included.

Instruments. Some of the metadata fields characterizing an instrument have already been defined when describing observation and state metadata. Therefore, these will not be described again.

Identifier. See observation / state metadata.

List of Instruments Constants. For each constant used to characterize the instrument, the following items must be clearly stated:

Type. The constant may be either a scalar or a matrix. In the case of matrices, their dimensions must also be specified.

Referencing. See observation / state metadata.

Units, Default Covariance, Scale Factors. See observation / state metadata.

Models. When describing a model in ASTROLABE, the specification of its *signature* (input and output parameters) is given. In other words, the description of models exactly corresponds to the structure of observation equations (see section 2.1,) where, besides the model identifier, the list of observation, instrument and state identifiers are provided. The metadata items that must be specified are:

Identifier. Unique code identifying the model.

List of States. The identifiers pointing to the states that will be needed by the model. Each state includes a sub-item, namely the role, to define whether the state will be just read (*constant* role) or estimated (*free* role.)

List of Observations. The identifiers of the observations used by the model to derive one or more states.

List of Instruments. The identifiers of the instruments that intervene in the estimation of the output states. This list is optional (as it is when writing observation equations.)

Time. All the time stamps in ASTROLABE datasets refer to the same coordinate reference frame and use the same units for simplicity reasons. A single specification for time stamps is therefore needed. The items to describe are:

Units. See observation / state metadata.

Referencing. See observation / state metadata.

Figure 1 depicts the actual specification of metadata for an observation (lines 1–28, l_spec XML tag,) a state (lines 29–52, p_spec XML tag,) a model (lines 53–72, m_spec XML tag,) and an instrument (lines 73–105, i_spec XML tag.) No metadata for time (stamps) has been included in this example.

3 THE FILE AND NETWORK INTERFACES

The data model described in section 1 has been materialized in two different interfaces, namely the file and network ones.

3.1 The File Interface

The first materialization of the ASTROLABE data model uses files to store datasets. There are several kinds of files included in this interface. The most important ones are observation, instrument and state data files as well as metadata files (see section 2.2.)

Observation files contain a sequenced series of observation and observation equations *records*. In the context of the file interface, observations are referred to as ”l-records” (due to the event marker, ”l”, used to tell them apart of observation equations.) Observation equations are referred to as ”o-records.” Figure 2 depicts a text, XML materialization of an observations file (note that line numbers are not part of the file.)

In this example, the event tag corresponds to the XML tag name; therefore, <l> tags describe l-records (observations) while <o> tags correspond to o-records (observation equations.) Only one of the records in the figure show the active flag. It is the ”s” (status) attribute (see line 1.) It may take two values, ”a” (active) or ”r” (removed.) If the active flag is omitted, the l- or o-record is assumed active.

Both the observation and model identifiers used in observations and observation equations are represented by the ”id” attribute. For instance, in line 1 the observation identifier is ”baro1”. The observation equation in line 6 involves the model whose identifier is ”pva1_d”.

In this dataset the observations collected by two different barometers of the same type are shown. The observations in lines 1, 10 and 19 correspond to a kind of observation whose identifier is ”baro1,” so they share the same type. But the ones in lines 1 and 19 come from one specific barometer whose instance identifier or ”n” attribute is 32, while the observation in line 10 was obtained from a different one, with an instance identifier equal to 33.

The observations related to (temperature compensated) barometers have one tag each (the temperature) whose values are 23.44, 23.45 and 23.45 respectively. The remaining kind of observations have no tags. In all cases, the covariance matrix includes the standard deviations only (so the assumption is that correlations are equal to 0 in all cases.)

Three different models are used by means of observation equations. See for instance lines 6 to 8, where three observation equations may be observed. There, it may be seen that the models used are those whose identifiers are, respectively, ”pva1_d”, ”imu1_bias_d” and ”height_update.” The observation equation in line 6 involves two states whose instance identifiers are respectively 27 and 11. To do it, it uses only an observation of type ”imu1” whose instance identifier is 41; no instruments are included in this equation.

Instrument Files. Contain the data defining the characteristics of the different instruments used in a dataset. Since the structure of the instrument data entity is the same than the one used for observations, l-records are employed to materialize instrument information. Thus, an instrument file contains a series of l-records. Although no example is provided to depict such files, Figure 2, showing observations, may be used as reference. Note, though, that o-records do not exist in instrument files.

State Files. The output, estimated trajectory which consists of a sequenced series of states, implemented as l-records once more. No o-records are present in state files.

Metadata Files. These are used to fully characterize the different entities integrating the ASTROLABE data model. See Figure 1 for an actual example of a metadata file.

Metadata files are always stored in XML format; on the contrary, data files may be stored in either text (XML) or binary formats. Binary files are recom-

```
<l_spec s="a">
  <type> baro </type>
  <lineage>
    <id>   baro1 </id>
    <name> Baro measurements </name>
  </lineage>
  <dimension> 1 </dimension>
  <ref>
    <ref_frame_VC>
      QNH
    </ref_frame_VC>
    <coor_system_VC>
      cartesian
    </coor_system_VC>
  </ref>
  <units> hPa </units>
  <c> 1 </c>
  <s> 1 </s>
  <t_spec>
    <dimension> 1 </dimension>
    <ref>
      <coor_ref_frame_VC>
        Celsius
      </coor_ref_frame_VC>
    </ref>
    <units> C </units>
  </t_spec>
</l_spec>

<p_spec s="a">
  <type> pva </type>
  <lineage>
    <id> pva1 </id>
    <name>
      Position, velocity and
      attitude in WGS84
    </name>
  </lineage>
  <dimension> 9 </dimension>
  <ref>
    <ref_frame_VC> WGS84 </ref_frame_VC>
    <coor_system_VC>
      geodetic, geodetic, geodetic,
      Lned, Lned, Lned,
      Bfrd-Lned, Bfrd-Lned, Bfrd-Lned
    </coor_system_VC>
  </ref>
  <units>
    rad, rad, m,
    m/s, m/s, m/s,
    rad, rad, rad
  </units>
</p_spec>

 <m_spec s="a">
  <type> RW_6_d </type>
  <lineage>
    <id> imu1_bias_d </id>
    <name>IMU biases as random walk</name>
  </lineage>
  <l_list>
    <dimension> 1 </dimension>
    <item n="1">
      <id> imu1_bias_pn </id>
    </item>
  </l_list>
  <p_list>
    <dimension> 1 </dimension>
    <item n="1">
      <id> imu1_bias </id>
      <role> free </role>
    </item>
  </p_list>
</m_spec>

<i_spec s="a">
  <type> baro_p0_h0 </type>
  <lineage>
    <id> p0_h0 </id>
    <name>Initial pressure+altitude</name>
  </lineage>
  <c_list>
    <dimension> 2 </dimension>
    <item n="1">
      <type> scalar </type>
      <ref>
        <ref_frame_VC> QNH </ref_frame_VC>
        <coor_system_VC>
         pressure
        </coor_system_VC>
      </ref>
      <units> mBar </units>
      <c> 1.2 </c>
      <s> 1 </s>
    </item>
    <item n="2">
      <type> scalar </type>
      <ref>
        <coor_ref_frame_VC>
          WGS84-ellipsoidal-height
        </coor_ref_frame_VC>
      </ref>
      <units> m </units>
      <c> 0.15 </c>
      <s> 1 </s>
    </item>
  </c_list>
</i_spec>
```

Figure 1: Metadata: observation, state, model and instrument.

mended when big amounts of data need to be handled; these are more efficient in terms of space and processing time. l- and o-records also exists in the binary version of data files; obviously, these are represented in a different (more compact) way.

3.2 The Network Interface

The second materialization of the ASTROLABE data model is the network interface, which relies on TCP / IP sockets. This interface implements, at least up to

```
<l id="baro1" s="a"  n="32"> 124.88 23.44 1023.44
                                   0.3                              </l>
<l id="imu1"         n="41"> 124.88       0.01 0.02 0.015 0.32 0.43 9.95
                                   1e-3 1e-3 1e-3  1e-2 1e-2 1e-2 </l>
<l id="imu1_bias_pn" n="17"> 124.88       0.0  0.0  0.0   0.0  0.0  0.0  </l>
<o id="pva1_d"             > 124.88       27   11   41                   </o>
<o id="imu1_bias_d"        > 124.88       11        17                   </o>
<o id="height_update"      > 124.88       27   34   32    51             </o>

<l id="baro1"        n="33"> 124.90 23.45 1023.45
                                   0.3                              </l>
<l id="imu1"         n="41"> 124.90       0.01 0.01 0.014 0.33 0.44 9.95
                                   1e-3 1e-3 1e-3  1e-2 1e-2 1e-2 </l>
<l id="imu1_bias_pn" n="17"> 124.90       0.0  0.0  0.0   0.0  0.0  0.0  </l>
<o id="pva1_d"             > 124.90       27   11   41                   </o>
<o id="imu1_bias_d"        > 124.90       11        17                   </o>
<o id="height_update"      > 124.90       27   35   33    52             </o>

<l id="baro1"        n="32"> 124.92 23.45 1023.45
                                   0.3                              </l>
<l id="imu1"         n="41"> 124.92       0.01 0.01 0.013 0.30 0.42 9.95
                                   1e-3 1e-3 1e-3  1e-2 1e-2 1e-2 </l>
<l id="imu1_bias_pn" n="17"> 124.92       0.0  0.0  0.0   0.0  0.0  0.0  </l>
<o id="pva1_d"             > 124.92       27   11   41                   </o>
<o id="imu1_bias_d"        > 124.92       11        17                   </o>
<o id="height_update"      > 124.92       27   34   32    51             </o>
```

Figure 2: An observations file in text (XML) format.

the moment, a subset of the data model. Only the transmission of observation data is available nowadays. This means that two software components sharing ASTROLABE data through the network interface *must agree* on the metadata characterizing the information to avoid misunderstandings.

The network interface defines a very reduced set of messages to exchange information; these comprise the l-record, o-record, end-of-data and acknowledgment-of-reception messages. Obviously, the information conveyed by the l- and o-record messages correspond to that included the l- and o-records described in the file interface. In spite of being so limited, the network interface has proven to be very useful in real-time systems, where different software modules cooperate to estimate a trajectory; the data collector components may exchange data easily with the component(s) responsible for the actual computation. No intermediate files are needed to implement such architecture.

4 THE ASTROLABE LIBRARY

The CTTC has implemented a portable C++ library composed of reader, writer (file interface), sender and receiver (network interface) classes. In short, it includes all the necessary tools to process ASTROLABE data and metadata. This comprises dealing with the different formats (text versus binary, observations versus other kind of files, for example,) or the ability to read observation files either in forward or backward directions. This is specially important for TDSs, since one typical approach to estimate trajectories is to process data in forward direction first, in backward direction then, and finally combine the two solutions thus obtained to derive a final trajectory.

Performance is another issue that has been addressed in this library. All readers and writers in the file interface implement a technique known as ”buffered reading (writing)” to reduce the amount of input / output operations, thus increasing speed. The use of this technique doubled the performance of CTTC’s TDS, NAVEGA. Finally, the network interface always transmits data in binary format for efficiency reasons.

5 ASTROLABE IN REAL-LIFE

The ASTROLABE data model, through its two interfaces and library, has been put to the test in several projects since 2009 (see the Acknowledgements section.) The contact with real-life situations helped to improve ASTROLABE so it became what it is nowadays. Section 2.1 states that any kind of observable with error distributions with first and second moments may be supported by ASTROLABE. This statement, which is true from a conceptual standpoint, is reinforced by the successful results obtained in real situations where ASTROLABE has been used. Here, a list of the observations modelled for the aforementioned projects is presented: GNSS raw data, GNSS position (x, y, z) IMU (linear accelerations, angular velocities,) EGNOS corrections, magnetic fields, pressure, time tagged distances, coordinates of tie points, ranges and angles.

6 CONCLUSIONS

For several years now, the ASTROLABE data model and its file and network interfaces, materialized in an Interface Control Document (ICD) (Parés et al., 2016) and in a portable C++ library, have been put successfully to the test in the field of trajectory determination systems. Real-life projects incorporating different kind of sensors and observations have served to improve it and validate it. The genericity and extensibility goals have been achieved, so change and innovation challenges may be properly faced at no software maintenance costs. ASTROLABE exposes a terse, compact interface, simple but powerful enough to make it practical in the specific field it has been targeted at: data for TDSs.

The authors are considering putting ASTROLABE in the public domain.

ACKNOWLEDGEMENTS

The research reported in this paper started around seven years ago at the former Institute of Geomatics and has been funded by means of several European projects. We would like to highlight ENCORE (Silva et al., 2011; Colomina et al., 2012), IADIRA (Silva et al., 2006), CLOSE-SEARCH (Molina et al., 2012), GAL (Skaloud et al., 2015) and ATENEA (Fernández et al., 2011).

REFERENCES

ASPRS (2009). LAS Specification. Version 1.3 R11. Available online: http://www.asprs.org/a/society/committees/standards/LAS_1_3_r11.pdf. Accessed: 2016-11-22.

Colomina, I. (1992). Discrete Mathematical Techniques in the Analysis and Adjustment of Hybrid Networks. In *XVIIth International Congress of the ISPRS*, Washington DC, USA.

Colomina, I., Miranda, C., Parés, M. E., Andreotti, M., Hill, C., da Silva, P. F., Silva, J. S., Parés, T., Galera, M. J. F., Camargo, P. O., Fernández, A., Palomo, J. M., Moreira, J., Streiff, G., Granemann, E. Z., and Aguilera, C. (2012). Galileo's Surveying Potential. *GPS World*, 23(3).

Colomina, I. and Molina, P. (2014). Unmanned Aerial Systems for Photogrammetry and Remote Sensing: a Review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 92:79–97.

Crippa, B., de Haan, A. A., and Mussio, L. (1989). The Formal Structure of Geodetic and Photogrammetric Observations. In *Tutorial on Mathematical Aspects of Data Analysis, ISPRS, Intercomission WG III/VI*, Pisa, Italy.

Elassal, A. A. (1983). Generalized Adjustment by Least Squares (GALS). *Photogrammetric Engineering and Remote Sensing*, (49):201–206.

Fernández, A., Diez, J., de Castro, D., Dovis, F., Silva, P., Friess, P., Wis, M., Colomina, I., Lindenberger, J., and Fernández, I. (2011). ATENEA: Advanced Techniques for Deeply Integrated GNSS/INS/LiDAR Navigation. In *24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS)*, pages 2395–2405, Portland, Oregon, USA.

Groves, P. D., Wang, L., Walter, D., Martin, H., and Voutsis, K. (2014a). Toward a Unified PNT Part 1, Complexity and Context: Key Challenges of Multisensor Positioning. *GPS World*, 25(10):18–49.

Groves, P. D., Wang, L., Walter, D., Martin, H., and Voutsis, K. (2014b). Toward a Unified PNT Part 2, Ambiguity and Environmental Data: Two Further Key Challenges of Multisensor Positioning. *GPS World*, 25(11):18–35.

GSA (2015). GNSS market report 4. Technical report, European GNSS Agency, Prague, Czech Republic.

IGS (2013). RINEX. The Receiver Independent Exchange Format. Version 3.02. Available online: ftp://igs.org/pub/data/format/rinex302.pdf. Accessed: 2016-11-22.

Karpowicz, J. (2016). Above the Field with UAVs in Precision Agriculture. Technical report, Commercial UAV Expo, Las Vegas, USA.

Kresse, W. and Fadaie, K. (2004). *ISO Standards for Geographic Information*. Springer, Berlin and Heidelberg, Germany.

Marietta, D., Smearcheck, M., and Racket, J. (2015). SPIDER and FLY: Navigation Data Simulation and Post-Processing Software Suite. In *ION GNSS*, Tampa, USA.

Minh, V. T. (2014). Trajectory Generation for Autonomous Vehicles. In *Mechatronics 2013: Recent Technological and Scientific Advances*, pages 615–626. Springer.

Molina, P., Colomina, I., Vitoria, T., Silva, P. F., Skaloud, J., Kornus, W., Prades, R., and Aguilera, C. (2012). Searching Lost People with UAVs: The System and Results of the CLOSE-SEARCH Project. In *International Archives of the Photogrammetry, Remote*

Sensing and Spatial Information Sciences, volume XXXIX-B1, pages 299–306, Melbourne, Australia.

OGC (2007). OpenGIS® Sensor Model Language (SensorML) Implementation Specification. Available online: http://portal.opengeospatial.org/files/?artifact_id=21273. Accessed: 2016-11-22.

OGC (2013). Geographic Information Observations and Measurements. Available online: http://portal.opengeospatial.org/files/?artifact_id=41579. Accessed: 2016-11-22.

Parés, M. E. and Colomina, I. (2015). On Software Architecture Concepts for a Unified, Generic and Extensible Trajectory Determination System. In *ION GNSS*, Tampa, USA.

Parés, M. E., Navarro, J. A., and Colomina, I. (2016). ASTROLABE. Interface Control Document (in preparation). Technical report, CTTC & Geonumerics, S. L., Barcelona, Spain.

Sarjakoski, T. (1988). Object-oriented Approaches in the Design of More Capable (Adjustment) Systems. In *XVIth International Congress of the ISPRS*, Kyoto, Japan.

Silva, P., Silva, J., Caramagno, A., Wis, M., Parés, M. E., Colomina, I., Fernández, J., Diez, J., and v. Gabaglio (2006). IADIRA: Inertial Aided Deeply Integrated Receiver Architecture. In *19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)*, pages 2686–2694, Fort Worth, USA.

Silva, P., Silva, J., Peres, T., Colomina, I., Miranda, C., Parés, M. E., Andreotti, M., Hill, C., Galera, J., Camargo, P., Diez, J., Palomo, J. M., Barbin, S., Moreira, J., Streiff, G., Granemann, E., and Aguilera, C. (2011). ENCORE: Enhanced Galileo Code Receiver for Surveying Applications. In *24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011)*, pages 3679–3689, Portland, USA.

Skaloud, J., Colomina, I., Parés, M. E., Blázquez, M., Silva, J., and Chersich, M. (2015). Progress in Airborne Gravimetry by Combining Strapdown Inertial and New Satellite Observations Via Dynamic Networks. In *IUGG 2015*, Prague, Czech Republic.

Soloviev, A., Veth, M., Yang, C., and Miller, M. (2016). Plug and Play Sensor Fusion for Navigation in GNSS-challenged Environments. *Inside Unmanned Systems*, pages 58–65.

Soloviev, A. and Yang, C. (2013). Reconfigurable Integration Filter Engine (RIFE) for Plug-and-Play Navigation. In *ION GNSS+*, Nashville, USA.

Titterton, D. H. and Weston, J. L. (2004). *Strapdown Inertial Navigation Technology*. The American Institute of Aeronautics and Astronautics & The Institution of Electrical Engineers, Reston, USA & Herts, UK, 2nd edition.

Tscherning, C. C. (1978). Defining the Basic Entities in a Geodetic Data Base. *Bulletin Géodesique*, (52):85–92.

Zhu, L., Yang, A., Wu, D., and Liu, L. (2014). Survey of Indoor Positioning Technologies and Systems. In *Life System Modeling and Simulation: International Conference on Life System Modeling and Simulation (LSMS 2014) and International Conference on Intelligent Computing for Sustainable Energy and Environment (ICSEE 2014)*.