A distributed computing environment for ATC data analysis

Karine Puechmorel, Daniel Delahaye

▶ **To cite this version:**

Karine Puechmorel, Daniel Delahaye. A distributed computing environment for ATC data analysis. DASC 2002, 21st Digital Avionics Systems Conference, Oct 2002, Irvine, United States. pp 2E4-1 - 2E4-9 vol.1, 10.1109/DASC.2002.1067930 . hal-00938411

HAL Id: hal-00938411

https://enac.hal.science/hal-00938411

Submitted on 5 May 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A DISTRIBUTED COMPUTING ENVIRONMENT FOR ATC DATA ANALYSIS

Karine Puechmorel, ENAC Dept. MI, 31055 Toulouse, France

Daniel Delahaye, CENA, 31055 Toulouse, France

Introduction

ATC data exchange is taking on increasing importance in both operational and experimental contexts.

From the operational point of view, information sharing between actors in the ATC system is the key to congestion reduction and safety. Data exchange allows traffic prediction, thus with an adequate model, the expected delay on flights may be obtained. With this value one may perform flight planning so that congestion is reduced and in turn delay itself. While this concept has been validated in the Collaborative Decision Making (CDM) project, there is a need for a global system allowing communication from any part of the world. Since data are generally expressed in different coordinate systems, it is necessary to perform conversions before being able to process them. It would be a great interest to have a built-in conversion process so that anybody willing to retrieve flight data may do that without caring about the referential in which data have been acquired.

Closely related to the problem of congestion reduction through flight planning is the computation of complexity metrics ([1], [2]). Those values will help the ATC system in finding out which area of airspace controllers must care about. Knowing that may lead to either better load balancing between control sectors or to designing new routes for aircraft.

On the other hand, designing future control aids requires extended experimentation and data analysis. Unfortunately, no standard exists for retrieving relevant information and this slows down the development process in many cases. The starting point of our work was precisely for dealing with different data formats between a variety of software. The problem of data exchange in simulation is complicated by the fact that in some experiments, real traffic is merged with simulated traffic, or simulation scenarios may change at a given time. This requires that the data storage system be able to unambiguously distinguish between simulation parts.

This paper addresses this problem through the use of a space-time distributed database that works in conjunction with a differential geometry library and a space-time location protocol. Once a user connects to the database, he specifies a time-space referential, which describes a projection map for earth coordinate representation and a time zone. All subsequent requests will be made implicitly using this referential from the user point of view, while data may be stored and processed internally in another referential. Each connection is thus associated with a conversion context object that performs the actual transformations. Since the client accesses data always by making queries through this object, conversion is made transparent.

A second point that has been treated is the time-space location problem: by using an octree structure on both spatial dimension and time, it is possible to find a hierarchical time-space partition in which event retrieval is fast. Within this frame, several individual ATC databases may be gathered in a distributed computing environment.

Combining the two features that are automatic conversion and space-time partitioning and database assignment, it is possible to construct a fully distributed, referential independent database system. The remaining point to be treated is how to specify the time-space area of interest and conversion context. For that, we have introduced a resource descriptor, similar to the URI/URL used in Web communications. This new descriptor is named UEL (Uniform Event Locator) and allows specification of all relevant information in a single character string:

- Earth model (Sphere, WGS84, ...) and chart (Stereographic, Mercator, Lambert) used.

- Spatial area and time interval in which the events must lie.
- Whether data comes from real or simulation traffic.

The tools that have been developed for achieving this goal will now be described more thoroughly.

Differential Geometry Library

Manifolds

Differentiable manifolds are the natural objects for geographic information processing since they describe sets that are equipped with a collection of charts (a concrete example of such charts being geographic maps). A chart allows to locally represent the manifold as a vector space, in which classical calculus is defined. Amongst the useful computations that may be done on a vector space, differential calculus deserves some special attention since it governs all flight mechanical equations. Being able to extend differential calculus on manifolds, which are only locally vector spaces, is the great achievement of differential geometry. For that, one must be able to define the notion of speed vector regardless of the chart used for performing the computation: this will yield to the concept of tangent vector. Furthermore, one may in many cases add a Riemannian structure on the manifold, which allows the computation of arc length.

Mathematical Preliminaries

From the mathematical point of view, a class C^r manifold is a set M equipped with a collection of charts [3] which are couples: $(U_\alpha,\varphi_\alpha)_{\alpha\in I}$ with U_α a subset of M and φ_α a one to one mapping from U_α to an open subset O_α of a given banach space E. The subsets $(U_\alpha)_{\alpha\in I}$ must form a covering of M, that is:

$$\bigcup_{\alpha\in I} U_\alpha = M$$

The change of charts applications: $\varphi_\alpha\circ\varphi_\beta^{-1}$ must be of class C^r. This last requirement allows application of differential calculus within a chart and still to be able to change to another chart while retaining all differential properties up to order of derivation r.

The sphere $S=\{x\in R^3 \mid \|x\|=1\}$ with the collection of the two stereographic projections at respectively north and south poles is an example of a differentiable manifold. Note that one may use a wider set of charts (for example Lambert projections), but at least two charts must be used. Of course the ellipsoid is again a differentiable manifold, with properties close to that of the sphere (in fact, ellipsoid earth models are so close to spheres that in many cases, little error occurs from confusing both).

Figure 1 shows how two stereographic projections give rise to a differentiable manifold structure on the sphere.

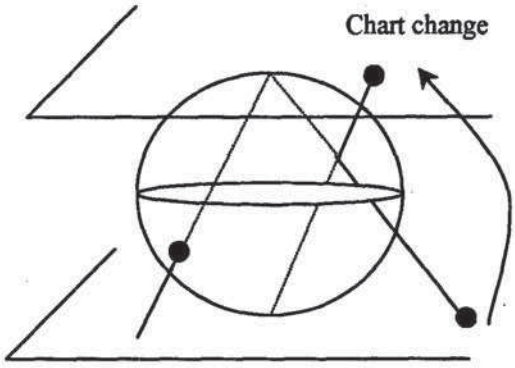


Figure 1. Stereographic Projections Giving Rise To A Differentiable Manifold Structure

Our implementation follows closely the mathematical definition.

First, the underlying abstract set must be defined. Since we will be dealing with infinite sets, it is not possible to store all points and use the standard implementations of the set. The only way of specifying sets will be by going back to the axiomatic definition. Sets are assumed to be elements of a collection called the universe (note that this collection will NOT be a set). Elements may be tested for fulfilling a property, and those for which the property is true form a set.

Implementation of that is more straightforward than it may seems at first sight, and is summarized in the UML diagram of Figure 2.

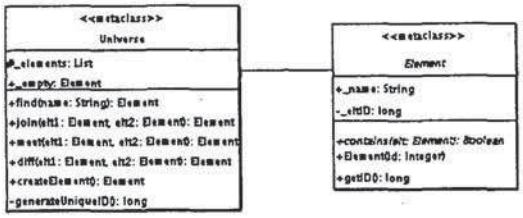


Figure 2. Set Class Diagram

The universe object is used as a container and factory for generating new instances of Elements. Elements are indeed logical sets, with standard meet, join and diff operations defined on them. The defining property of the set is specified by the "***contains***" operation that concrete sets must override (default behavior of "contains" is to return false for all but the _empty Elements). Each element has a name, that may be left null, in which case the element is anonymous, and a unique identifier (integer number) generated by the universe. Elements may be constructed from elementary Boolean operations: logical ***and*** for the meet, logical ***or*** for the join, logical ***not*** for the complementary set and so on.

The universe is responsible for the destruction of objects (garbage collection). Depending on the implementation language, this may require writing a garbage collector. We have developed both Java and C++ version of the library, the last one being much faster and ideally suited for heavy load applications. In this case, a standard mark-and-sweep algorithm performs garbage collection.

The Element class can act as the underlying set for the Manifold implementation. The other part of Manifold will be the set of charts. Implementing a chart is somewhat awkward since one must associate a subset of the manifold with the chart. Although our Element class allows an easy construction of subsets, it is convenient to be able to generate a defining property by enforcing the image of the subset to be a polygonal area. Of course, a chart is itself an instance of an Element so that manifold points may be tested for being in the domain of the chart. In fact, since the domains must cover the whole manifold, it is sufficient for the manifold to be viewed as a set of charts.

Any concrete chart must implement the following operations:

- Take an element of the manifold and compute its image as a vector if the point is in the domain of the chart, otherwise, raise a "NotInDomain" exception.
- Take a vector and find its inverse image as an element of the manifold if the initial vector is in the image of the domain of the chart, otherwise raise a "NotInCodomain" exception. The returned value is an Element that belongs to the manifold. The actual type of that Element is in fact a Manifold.Point, which is an inner class of manifold with some extra information added (in a concrete manifold, this is mainly the so-called intrinsic coordinates, like the latitudes and longitudes for spheres and ellipsoids).
- Compute the image of a vector by a change of map, raising a "DomainMismatch" exception if the operation cannot be completed.

Concrete Manifold for ATC Applications

Two concrete classes of Manifold have been derived, the sphere and the ellipsoid, which cover all our needs for geographic applications. Nineteen built-in earth models have been defined, the most useful being WGS84 and SPHERE (Standard radius of 6370997 meters). Amongst available charts, the most commonly used are stereographic,

Mercator and Lambert. A total of 30 different projections are implemented with their inverse. Manifold points are Elements with two real values for the latitude and longitude respectively. Since we are dealing with ATM, a third one is added for the altitude.

All manifold points are Elements, so may be named and retrieved by their name (in fact a hash table is associated with concrete manifolds so that the process of querying for a given point is fast). This feature allows an easy definition of beacons and reference points. Objects than are instance of earth models may read an XML [4] beacon file at creation and store the resulting points in the hash table. This allows entering known beacons for a country or area, which is under the responsibility of the database owning the object.

Some very usual projection maps may be defined similarly at creation time through an XML file. For example, most French radar information is stored in the Cautra IV coordinate system, which is a Sphere Lambert projection. Here is an extract of the file where Cautra IV is defined:

```
<node name="cautra4">
  <map>
    <entry key="Type" value="Lambert" />
    <entry key="Latitude" value="47" />
    <entry key="Longitude" value="0" />
    <entry key="extX" value="1e6" />
    <entry key="extY" value="1e6" />
  </map>
</node>
```

Tangent Vectors

Let M be a class C^r manifold with charts $(U_\alpha,\varphi_\alpha)_{\alpha\in I}$. A tangent vector to M at the point $x\in M$ is a class of equivalence among triples (U,φ,v) with (U,φ) a chart with $x\in U,\varphi(x)=0$ (such a chart is said to be centered at x) and $v\in E$ with respect to the equivalence relation:

$(U,\varphi,v)\approx(V,\psi,w)$

if

$$w=D(\psi\circ\varphi^{-1})_0(v)$$

In other words, a tangent vector represents a kind of speed vector, which may be expressed in different charts but with a formula for computing the different representations.

The implementation of the tangent vector class is straightforward:

- A manifold Point, which is the origin of the tangent vector.
- A chart centered at that Point (centering is tested when the constructor is called).
- A vector.
- A Boolean class method testing for equality (same equivalence class).

The tangent vector object is used for coding aircraft speed: since it is chart independent, correct format of retrieved and queried data is insured.

Library Features

The differential geometry library allows several usual operations and constructs, like differential calculus on manifolds. At the moment, it is far for being complete, but powerful tools are available for ATM applications:

- Riemannian manifolds implemented. This means that paths length may be computed. Geodesic curves between two manifold points may be constructed, but this feature is limited to spheres and ellipsoids.
- Simple lie algebra computations, like the lie bracket. This is important for trajectory planning, since the lie group of rotations/translations acts on the natural configuration space for the flight mechanics.

UEL Definition

UEL stands for "Uniform Event Locator" an acronym derived from the well-known URL [5]. It gives a unified view of events in space and in time and allows information processing in a referentially independent way. For example, French ATC data are mainly stored in CAUTRA 4 format, which is a Lambert projection, while some others are stored in latitude/longitude coordinates. The UEL formalism allows to make queries and to obtain results in a

referential specified by the user, regardless of the internal storage referential. We will now describe more accurately how an UEL in organized.

The Context

This is the first field in the UEL. It refers to the source of data, that is real traffic or simulation. The ATC context will always refer to a real situation, while other names may be used for storing and retrieving data from simulations. Each context is stored independently from the others in the database.

The Earth Model

This field is separated from the context by a colon ':' and indicates in which model of the earth the query will be done. This may be important for high accuracy or wide area computation, in which case the difference between a spheroid model and an ellipsoid model is significant. For now, only the standard spheroid and some ellipsoids (including WGS84) are implemented, since they will cover most of the users needs. However, introducing new models is simple, since the underlying implementation is that of a differentiable manifold.

The Sector Location

Airspace is divided in polygonal areas called sectors. Those polygons are almost always defined in local maps since their spatial extents are small enough to replace geodesics by straight lines. Elementary sectors may be gathered into control areas and ultimately into countries. This hierarchical structure is referred to in the third part of the UEL. Each element of the hierarchy is either:

- A named polygonal area. Existing sectors belong to that category.
- A specified polygonal area. Those are given as succession of vertices, which may be again named or specified. Specified polygons are versatile, but somewhat awkward to enter by hand. They are most useful in software generated UELs.
- A cluster of polygonal areas, which is again a polygonal area.

Optional Arguments

The last part of the UEL is similar in syntax to the optional arguments of a URL. It is separated by a question mark '?' and organized in pairs 'variable=value'. Amongst these, the following are very useful in practice:

MapType='projection type'. Specify a projection for the data. Expects to find a MapParameters='(list)' somewhere!

MapName='name'. Specify a map for the data by its name. Understandably, the name must be present in the system.

TimeZone='name'. Specify a time referential.

First='date', Last='date'. Filters the events so that only those occurring between those dates are taken into account.

UELs in Practice

Like URLs, UELs are primarily intended for information retrieval on a network of databases. Because data may be scattered over several machines, a resolution protocol has been defined. However, since search criteria are space and time locations, UEL resolution is based on servers storing polygonal managed area and time segments. Once a connection request is made to a specified UEL, the query is propagated upwards in the hierarchy of servers until one has a managed area containing the whole specified area (and also for the time segments). From this server, the query is processed downwards through the hierarchy to the physical databases. Unlike the case of name resolution which occurs in network protocols, things are made more complicated when the query area is not managed by a single database, in which case it must be split in several pieces, each associated with a physical database. Connections are established as 1 client to n servers, with $n \geq 1$.

Database Organization

Overall Design

A successful connection request through a UEL will first result in the creation of an instance of the conversion context object that will manage all incoming information preprocessing and outgoing

information post processing. Conversion contexts are uniquely associated with connections even if different requests use the same UEL, and are deleted when the connection is closed. Before the conversion context, all data is in the internal storage format (it is recommended that this format will be WGS84 latitude/longitude/altitude since this is the preferred choice for published space data).

Requests to the database are done with XML queries. Triggers may be entered so that the user will be notified by an XML message of some event occurring in the database. Common triggers include new tracks coming, new strips emitted, flight plan entered or updated, security alerts, etc.

Tables

The following tables are implemented in the current version:

- Flight plan information including departure and arrival beacons, departure time and estimated arrival time, aircraft type.
- Flight plans segments defined by beacons and estimated arrival time.
- Beacons defined by name and position. They are implemented as Points in the reference manifold (sphere or ellipsoid).
- Sector definitions as lists of beacons or named points. Sectors are always polygonal, with geodetic segments.
- Track events (Radar data) with time, position, speed vector (tangent vector to the reference manifold), aircraft identifier (must be unique in the database). Transponder code is optional.
- Security events (TCAS alerts).

Connection

Client Side

The following steps describe connections to the database from the client side:

- Make a UEL request to the UELResolver service.
- Upon completion of the resolving process, the UELResolver returns a set of connections with the physical network locations of the services responsible for the time-space area defined in the UEL. Note that this must be a set since a UEL may overlap several areas of responsibility. An empty set is returned if no databases can be found for managing the UEL. Note that an empty set is returned even if some databases have their responsibility area inside the UEL, but do not cover the whole of it. Malformed UELs or Communication Failures raise the corresponding exceptions.
- The elements of the set of connections are provided with all conversion procedures needed and are ready to be used. However, the physical connection process is submitted to late binding, that is, socket opening is delayed until a request is made in the responsibility area of the particular database addressed. On the other hand, connections are closed inside the pool after a user-defined timeout or in case of reaching the maximum number of connections (in this case, the least recently used is closed).
- When a request is made to a given UEL, the UELResolver finds the relevant connection and opens it if necessary, according to the late binding principle. Connections require an authentication with a pair user/password that is associated with rights granted on the database. For safety and confidentiality reasons, connections are opened using SSLSockets.
- Request returns are XML documents.
- Some requests create triggers on the server. Those procedures are fired when a given condition is met (in our experimental implementation, those are ATC events as defined in the simulation tool used). Triggers may be chained and several clients may be associated to event triggers. A trigger returns an XML document corresponding to the request made. Event triggers are useful for broadcasting database changes to interested clients. The traffic simulator that we have used for our experiments is based on message passing, the destination of messages being determined by matching a regular expression against the message body. Triggers offer the same functionality, but add a lot of new features, like the ability to interact with the database. Some event triggers are not accessible or customizable

for all users since this may result in a drastic reduction of performance. This is the case for radar track acquisition that occurs frequently. Allowing computer expensive requests to be processed on those events will overload the server when traffic is high. In order to avoid this, only authorized users may create event triggers with the custom query procedure. In any case, standard event triggers have the higher priority. Triggers programming and compilation are still under development, although most frequently used functionality is implemented.

- Connections remain open until the timeout expires or the connection set object is deleted. This choice has been made to favor speed at the expense of allowing fewer connections. When a client disconnects, a message is sent to all servers involved.

Server Side

From the server side, the connection is made to the connection manager service, which is in charge of processing incoming requests and dispatching them. Several physical databases may be under the control of a single connection manager and extended services may be made available, like trajectory prediction. Services under the responsibility of the connection manager are accessed via CORBA. The overall architecture is summarized in Figure 3.

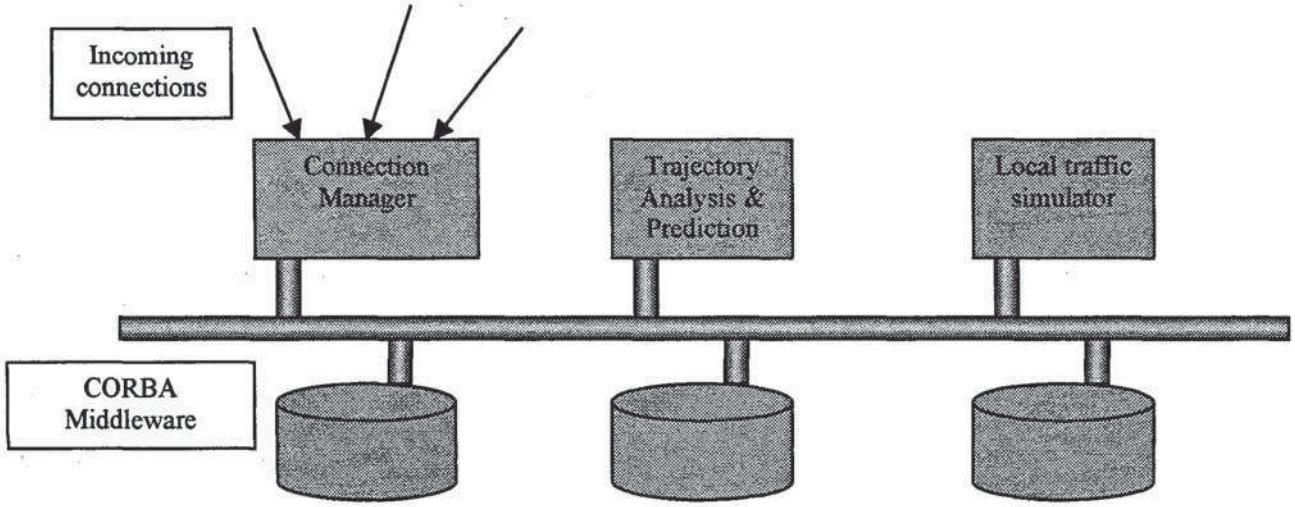


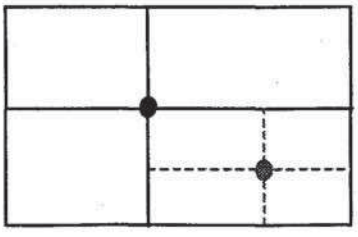
Figure 3. Services Communication

Space-Time Responsibility Domains

This is the core of the information retrieval system. Each database is in charge of some areas of the earth for given time segments. Efficient data access depends on setting up a fast search structure, adapted to this kind of information. Several structures exist for spatial segmentation, the more widely used being:

- Quad and Oct trees.
- Delaunay triangulations.

The first of these takes points and constructs a hierarchical decomposition of the space in rectangular cells. The construction of the tree is done by successive insertion of points: each time a point is inserted, the tree is searched for locating the cell in which it lies, then that cell is split into 4 subcells (or 8 for a 3-dimensional space) sharing the inserted point as common vertex. The procedure is summarized below.



The solid lines show the cell decomposition at the first stage, then the dashed ones show what happens after a new point insertion.

For our application, Oct trees must be used since we are dealing with 3-dimensional data (two dimensions for the spatial location and one for the time). Tree structure is essentially very fast for point location: this is of major importance in our application since UEL resolution and query processing is made through point and area location procedures. Oct trees must be kept as balanced as possible in order to allow fast processing. Furthermore, points must be chosen such that the granularity is fine enough to accurately represent the ATC sectors.

Top level UELResolvers are in charge of coarse subdivision, while individual databases have a list of elementary cells as their responsibility domain. Each cell may be linked to more than one UELResolver or database since responsibility domains may overlap. In the case of multiple databases linked to the same reference cell, the previous level UELResolver will send the whole list of databases eligible to process the requested UEL to the calling client. The actual spatial segmentation on that list is performed by API procedures on the client machine. This avoids unnecessary computation on the server machines, thus reducing the time-consuming tasks on those critical parts of the system.

Caching

Since most requests deal with events closely located in space and time, like trajectory retrieval, caching data is very important for sustained performance under heavy load conditions. However, things are made harder than usual because of the spatio-temporal nature of events. Most requests fall into two categories:

- Queries about parts of trajectories for a given set or aircraft.
- Queries about a given spatial area (generally for traffic analysis and monitoring).

This suggests the use of the following heuristic for caching data:

- New radar tracks are always cached and stored until the associated flights leave the responsibility area or the maximum memory depth has been reached.
- Each time a request is made on flights in a given area, all associated trajectories are fetched into memory, with a maximum length.
- Old trajectories (from the access time point of view) in cache are dumped to database when timing out or if memory is low.
- Cache is physically organized as a hash table based on the flight identifier (necessarily unique).

Because of the structure of space-time in cells, caching is done with elementary cells. This means that all events related to a given space-time area will be covered by such cells, then each cell will be fetched into the cache.

Overall performance with this procedure is very satisfactory: in fact, on our test system, the network communication was only a fraction of the total CPU time used by the ATC data generation and analysis system.

Further Work

The UEL resolving system has been used for about 8 months now and has proved workable. We are currently seeking to develop a air traffic navigator, based on the same concepts as Web Browsers. The idea is to allow the user to enter an UEL, then display a radar image of the area referenced by this UEL. Since all queries are made using XML, it is quite simple to implement Web services for accessing databases. Selecting a flight with a pointing device will result in information display about it, the level of accuracy depending on the user class. UEL navigators may thus be used on control positions as powerful tool for controllers.

The second way of improving our computing environment is by adding services extending data processing. Among them, it is planned to release a traffic simulator and a trajectory generation tool.

Conclusion

We have presented a distributed ATC database system that can be used to retrieve and process data that are spatio-temporal. The ability to specify areas

of interest by means of a unified descriptor, the UEL, allows the design of versatile ATM systems. Furthermore, since part of the project was the coding of a differential geometry library, many functions may be used for flight mechanics applications.

Due to the hierarchical structure, it is easy and fast to select databases responsible for a given UEL, so that the overhead induced by the protocol is nearly transparent for the user. The ATC database is in an early development stage, but it has been intensively tested for the computation of traffic complexity metrics.

References

[1] Sridhar, B., Seth, K.S., Grabbe, S., 1998, "Airspace complexity and its application in air traffic management", 2nd USA/EUROPE ATM R&D seminar, Orlando.

[2] Delahaye, D. and Puechmorel, S. 2000, *Air Traffic Complexity: Towards Intrinsic Metrics*, Proceedings of the Third USA/Europe Air Traffic Management R&D Seminar, Napoli, Italy, June.

[3] Berger, M. and Gostiaux, B. 1988, *Differential Geometry: Manifold Curves and Surfaces*. Springer-Verlag GTM.

[4] Bray, T., Paoli, J., Sperberg-McQueen, C.M., and Maler, E., *Extensible Markup Language 1.0* W3C Recommandation., http://www.w3.org/TR/REC-xml

[5] Berners-Lee, T., Masinter, L., and McCahill, M., *Uniform Resource Locators* (RFC1738) http://www.w3.org/Adressing/URL/Overview.html