INTEGRATING THE GALILEO APPLICATIONS FOR SIMULATION OF OFFSHORE SYSTEMS VIA THE GXML UNIFIED FORMAT

Ismael H. F. Santos
ismaelh@petrobras.com.br
CENPES, Petrobras, Rio de Janeiro, RJ, Brazil
Vanessa Braganholo
braganholo@dcc.ufrj.br
DCC/UFRJ, Rio de Janeiro, RJ, Brazil
Marta Mattoso
marta@cos.ufrj.br
PESC/COPPE/UFRJ, Rio de Janeiro, RJ, Brazil
Breno P. Jacob
breno@coc.ufrj.br
LAMCSO/PEC/COPPE/UFRJ, Rio de Janeiro, RJ, Brazil
Carl Albrecht
carl@deg.ee.ufrj.br
LAMCSO/PEC/COPPE/UFRJ, Rio de Janeiro, RJ, Brazil

Abstract. *Computer based simulations of engineering-related problems are computer and data intensive. In a typical scenario, data is usually passed from one program to another in order to complete several steps of the simulation. The sequence of operations to perform a simulation can be modeled as scientific workflows. However, this creates an interoperability problem, since in most of the cases, conversion steps are needed every time a different program needs to be run over the data. In the Galileo research project, this problem is crucial. The research team of Galileo consists of engineers, geologists, computer and computational scientists from Petrobras Research Centre (CENPES) and five Universities in Brazil developing innovative computational mechanics applications to face the new exploration and production challenges in recently discovered pre-salt areas in ultra deep waters (> 2,000 meters) southeast of Brazil. One of the main efforts in the Galileo project is to share scientific and engineering data in the context of scientific workflows. To solve this problem, a unified data format that must be shared by all Galileo applications have been defined, referred here as GXML (Galileo XML).*

Keywords: *GXML, XML Schema Definition, interoperability, simulation, offshore systems*

1 INTRODUCTION

Computer based simulations of engineering-related problems are computer and data intensive. In a typical scenario, data is usually passed from one program to another in order to complete several steps of the simulation. The sequence of operations to perform a simulation can be modeled as scientific workflows (Deelman et al., 2009; Oinn et al., 2004). This creates an interoperability problem, since in most of the cases, data conversion steps are needed every time a different program needs to be run over the data.

An ongoing research project where this problem requires a solution is the Galileo project, coordinated by CENPES/Petrobras. The research team consists of engineers, geologists and computational scientists from five Universities in Brazil (USP, COPPE/UFRJ, PUC-Rio, UFAL and ITA), developing innovative computational mechanics applications to face the new exploration and production challenges in recently discovered pre-salt areas in ultra deep waters (> 2,000 meters).

One of the main efforts in the Galileo project is to share engineering data in the context of *scientific workflows* that are being considered in design activities of floating production systems (FPS) (Carvalho et al, 2009), including for instance the design of the mooring system that keeps the FPS in a stationary position. Another example is the design of the *risers*, which are pipes that convey fluids between the platform and the wellhead on the sea floor. To certificate the operation of the risers for their entire life cycle, simulations are conducted under fatigue and extreme environmental conditions, based on meteo-oceanographic data on wind, wave and currents, to determine the stresses acting on the risers and compare them against design criteria. In (Carvalho et al, 2009) it is stated that the workflow for riser design comprises a sequence of simulations (hydrodynamic hull analyses, coupled motion analyses, riser structural analyses and fatigue analyses), performed using many applications that are being incorporated into the Galileo project.

These Galileo's applications have been using their own text-based data format, thus making interoperability among the applications difficult, and data files very large. As an example, Figure 1 shows selected sections of the data format used by one of Galileo's applications, Prosim (Jacob, 2006). This particular example corresponds to the coupled model of a semi-submersible platform, describing the hull, the mooring lines and risers. The complete file has 90492 lines and 10,3Mb.

```
$SITUA 1.7.0(3.1)
Modelo P52 - Boia Leste
NEWN  NLOADC 1 STSIM  DYSIM  DAMPC FATDI    /
$ Parametros de análise Dinamica
$ 6. DT, TOTALT, TSPAN, TPRNT, TCOS
0.1 10800.0 10800.0 400.0 400.0
$ 13. Parâmetros Globais - RHO, G, WVI, ROAR
1.025 9.806 0.000000924 0.001222
$ 15. NUF - Número de unidades flutuantes
    ... TRUNCATED LINES ...
$ 19. XSI_G, YSI_G, ZSI_G,PROA
0.00000000 0.00000000 -34.00000000 0.00000000
$ 20. Peso, FACTML, XCG, YCG, ZCG, TENMAS, TENMASZ
939878.6238 1000000 0.75000000 -1.04000000 31.71000000 0 0
$ - Linhas -
$ 40. NP1, NP2, NP3, NP4, NP5, NP6, NP7
0 0 0 0 0 0 54 0
$ 48. NNOMAX, NNOPX, NFDMAX, NFTMAX, NPFDMAX, NPFTMX
4185 21 2 1 2 2
```

Figure 1 – Example of data format used by the Prosim application

To solve the Galileo's data interoperability problem, a unified data format have been defined that will be shared by all Galileo applications. This format is called GXML (Galileo XML). As the name says, it is based on XML, which can be easily handled by applications

using standard XML APIs. Since the beginning, the use of XML brought some concerns that had driven the architecture of the proposed solution, such as:

(1) Scalability when dealing with large volumes of data;
(2) Seamlessly integration into host applications;
(3) Creation of an abstraction layer to isolate host applications from the details of XML file access and interpretation (XML parsing).

There are several data formats available (Hartnett & Rew, 2008; HDF Group, 2008; Ensight, 2009; Schroeder et al, 2006; Moreau et al., 2005; Caron et al., 2008; Clarke & Mark, 2007; Elias et al., 2009), but they all present limitations in addressing all the above concerns as required by the Galileo's applications.

In this paper, the GXML format is presented, together with details of the architecture and how it is going to be used within the context of Galileo to solve its interoperability problems. Scientific data interoperability is a problem that is bound to become more complex as new software solutions are developed. This paper can provide hints to similar projects in solving their interoperability problems.

The remaining of this paper is organized as follows. Section 2 presents a brief description of the Galileo project. Section 3 presents related work regarding formats for scientific and engineering data that has motivated the creation of the GXML format. A history of the building process and formal specification of GXML is presented in section 4. Section 5 presents further details of the GXML architecture. Finally, conclusions and suggestions of future works are presented in section 6.

2 BACKGROUND: THE GALILEO PROJECT

Galileo is a Collaborative Problem Solving Environment (CPSE) for Offshore Engineering projects, tailored for assisting the control and execution of engineering projects in the oil, gas and energy industry. Its main objectives are to apply state-of-the-art techniques of scientific workflows and 3D immersive visualization for multi-physics and multi-scale engineering problems (Santos et al., 2008b).

The creation of Galileo is motivated by the necessity of finding effective solutions for collaboration of team workers during the execution of large and complex Offshore Engineering projects. Those projects usually require the execution of a large number of engineering simulations, encapsulated as engineering services, combined in different orders and rearranged in different subsets according to project requirements. By means of a Scientific Workflow Management System (SWfMS) users are able to orchestrate the execution of engineering simulations as workflow tasks that can be arranged in many different ways. The most interesting cases can also be selected for visualization to allow the engineers to better understand the simulated phenomena. This is frequently the last step of the workflow that represents the simulation experiment (Santos et al., 2008a).

A Problem Solving Environment (PSE) is a specialized software system that provides all the computational facilities needed to solve a target class of problems. These features include advanced solution methods, automatic and semiautomatic selection of solution methods, and ways to easily incorporate novel solution methods. Moreover, PSEs use the language of the target class of problems, so users can run them without specialized knowledge of the underlying computer hardware and software technology.

Collaborative Problem Solving Environments (CPSE) focus on the development of a PSE coupled with collaborative environments to support the modeling and simulation of complex scientific and engineering problems. These capabilities enable engineers to easily setup computations in an integrated environment that supports the storage, retrieval, and analysis of the rapidly growing volumes of data produced by computational studies.

Galileo, as a proposed CPSE, should allow users create their own workflows combining different sequences of applications in order to solve offshore engineering problems. Therefore the data integration is a first step towards the development of a fully integrated simulator.

3 FORMATS FOR SCIENTIFIC AND ENGINEERING DATA

There are several available data formats to represent large scientific and engineering data. Lots of them use binary representation, such as NetCDF (Network Common Data Form) (Hartnett & Rew, 2008), HDF5 (Hierarchical Data Format) (HDF Group, 2008) and Ensight Gold (Ensight, 2009). This is a good choice when space requirements are critical, since binary formats are more compact than text-based formats. However, binary data is not self-descriptive, and thus interoperability is seriously compromised.

To try to overcome this limitation, text/XML-based formats, such as VTK/XML (Schroeder et al, 2006) and XDTM (Moreau et al., 2005) arise. On the other hand, while solving the interoperability problem, the space requirements increase and the time needed to send a huge text-based file from one machine to another can be unacceptable.

Some other formats mix binary data with XML description, including the following:

- NcML (NetCDF Markup language) (Caron et al., 2008) is an XML representation of netCDF metadata. This metadata corresponds (approximately) to the header information of a netCDF file. Thus, the information itself is stored in the netCDF file, while metadata that describes its structure is stored in the associated XML file.
- XSIL (Blackburn et al., 1999), an acronym for Extensible Scientific Interchange Language, represents (meta) data for scientific computing. It was developed for astronomical and gravitational wave communities but provides general purpose tags for scientific data. References to binary data are allowed in the context of the XSIL file.
- XDMF (Clarke & Mark, 2007; Elias et al., 2009) classifies data as light and heavy according to the amount of information it represents. In this sense, light data is allowed to be stored in the XDMF file's body, while heavy data is stored in HDF5 format and described in the XDMF file.

While the aforementioned formats are capable of representing metadata in XML format, they are different in the sense that XDMF and XSIL allow light data to be stored in the XML file itself. This is not true for NcML, since all data are actually stored in the NetCDF file.

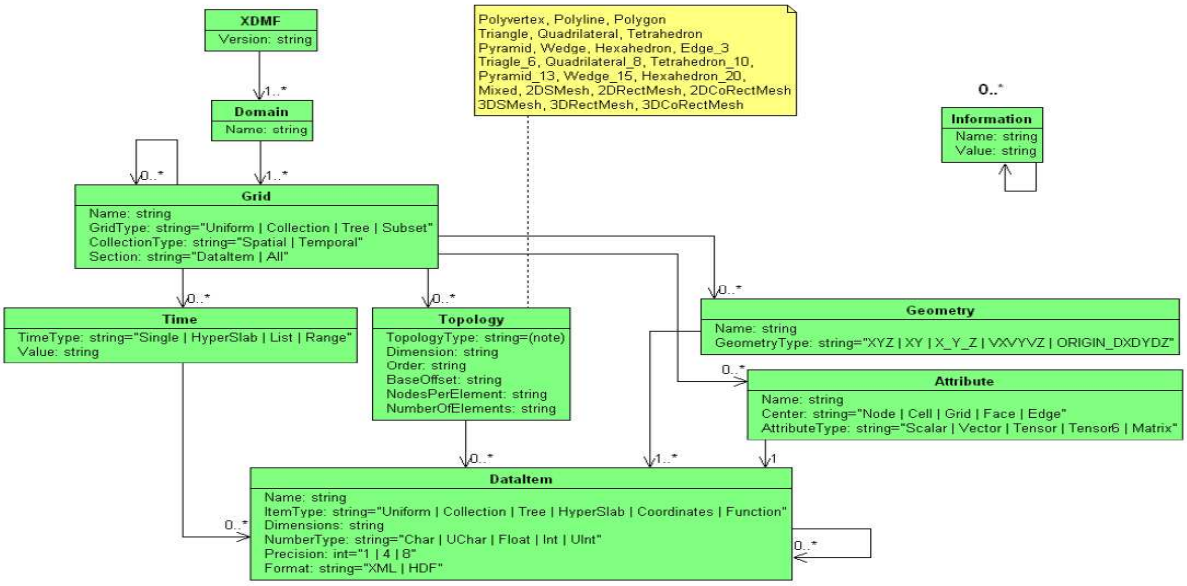

Figure 2 – A representation of the XDMF structure

XDMF was developed in a standardized way to exchange scientific data between high performance computing (HPC) codes and tools. It has been used as both a common format for visualization of many different codes and as a vehicle for developing coupled simulations. From Galileo's perspective, XDMF was a good starting point, because it provides interoperability and compact data at the same time. For this reason, the remainder of this section presents some details of XDMF.

As mentioned before, XDMF stores light data directly in XML format, while heavy data is described in XML but actually stored in HDF5 files. The data format is stored redundantly in both XML and HDF5. This allows tools to parse the XML document to determine the resources that will be required to access the heavy data. A C++ application programming interface (API) is provided to read and write XDMF data. This API has also been wrapped so it is available from popular languages like Python, Tcl, and Java.

Figure 2 shows a graphical representation of the XDMF structure. It is capable of representing one or more Domains. Domains are composed of Grids, each of which represents a collection of homogeneous elements. Each Grid can have temporal data (Time), other Grids, descriptions of the data organization (Topology), XYZ mesh values (Geometry) and mesh values (Attribute). Data itself is stored in lower level elements called DataItem. There is also a special element called Information that can store application specific data.

Figure 3 shows an example of XDMF file. Notice that it is an XML document, with **elements** (such as <Domain>) and **attributes** (such as Name="EdgeCFD"). Elements can contain text or other elements, while attributes are single valued. As an example, in the figure the Domain element contains a Grid element. Lines shown in green in the figure represent comments, and are not part of the content of the file.

```
<XDMF xmlns:xi="http://www.w3.org/2001/XInclude" Version="2.0">
  <Domain Name="EdgeCFD">
    <Grid Name="foo_00000" GridType="Collection" CollectionType="Spatial">
      <Time TimeType="Single" Value="0.00"/>
      <!-- GRID FOR THE FIRST PARALLEL (SUB)DOMAIN -->
      <Grid Name="foo_000" GridType="Uniform">
        <Topology Type="Tetrahedron" NumberOfElements="   296882 " BaseOffset="1">
          <DataItem Dimensions="1187528 " NumberType="Int" Format="HDF">
             foo_000_00000.h5:/incid  <!-- MESH INCIDENCE -->
          </DataItem>
        </Topology>
        <Geometry Type="XYZ">
          <DataItem Dimensions="1472283" NumberType="Float" Precision="8"
                                                           Format="HDF">
             foo_000_00000.h5:/cords  <!-- NODAL COORDINATES -->
          </DataItem>
        </Geometry>
        ... TRUNCATED LINES ...
      </Grid>
      <!-- GRID FOR THE SECOND PARALLEL SUBDOMAIN -->
      ... TRUNCATED LINES ...
   </Grid>
  </Domain>
</XDMF>
```

Figure 3 – Example of a XDMF file

HDF5 files can be referenced inside *DataItem* elements. In Figure 3, different portions of an HDF5 file are referenced. As an example, the first DataItem element points to the *incid* array inside the ***foo_000_00000.h5*** file (this is specified by the *DataItem* value ***foo_000_00000.h5:/incid***). Notice that the explicit declaration of the metadata information in the XDMF facilitates the data access of HDF5 files.

4 BUILDING THE GXML FORMAT

This section presents the history and specification of the GXML format, starting by gathering the data requirements of Galileo's applications (Section 4.1) in order to build the GXML Domain Model, which represents the relationships among the main elements in an offshore engineering simulation domain. Then, an explanation is given regarding why the Galileo team chose to be compatible to XDMF while extending it to represent the required data (Section 4.2). Section 4.3 discusses the GXML format and its representation using XML Schema Definition, and also presents some XSLT transformations created for visualizing GXML data. The concepts described here are illustrated with some case studies related to the simulation of floating production systems.

4.1 First Stage: Preliminary Specification of Contents and Structure

The building of the Galileo unified data format began with the specification of the contents and structure of the data to be included in the unified data format. This goal was accomplished during an intensive phase in the development process when several meetings were conducted between the members of the different teams involved in the project.

After a few months, a preliminary specification was obtained, including the description of global data; platform data; risers and mooring lines data, and data for other scalar components of floating offshore systems such as buoys, springs, concentrated loading and masses. The specification included also environmental loading data of waves, wind and current, and analysis parameters. The complete specification is very extensive and would fill dozens of text pages, and therefore will not be presented here. Figure 4 illustrates a small section that corresponds to some global data related to the sea bottom.

```
GLOBAL_DATA   {
   GLOBAL_SEA_BOTTOM    {
         BATHIMETRY_INFO  {
                  TYPE of BATIMETRY_FILE  < sgo-dgn | dat | mg >
                  BATHIMETRY_FILE - string
                  LEVEL_CURVE   {
                           Z  -float,     NR_POINTS - Integer
                           < X, Y >  -float-float
                  }
         HEIGHT_MAP    {
                           XMIN, YMIN-float-float  NR_POINTSX, NR_POINTSY- Integer
                           DIMENSION_X, DIMENSION _Y  -float-float
                           <FLAG_VAL_INFO- Integer, PROF, float PROPERTY_INDEX-Integer >
                  }
         }
         PLANE_INFO   {
                  PLANE_POINTS  { p1- Point3D, p2- Point3D, p3- Point3D }  // No collinear points
         }
         PHYSICAL_PROPERTIES {
                  NORMAL_STIFFNESS –float,        FRICTION_KEY  - Integer
                  FRICTION_PARAMETERS { deseax, deselt, fcax, fclt } - float float,float,float
         }
   } end GLOBAL SEA BOTTOM
   ... TRUNCATED LINES ...
} end GLOBAL DATA
```

Figure 4 – Sections of the Preliminary Specification

The simplified GXML Domain Model depicted in the Figure 5 describes the relationships amongst the main data groups, encompassed in a Scenario which might represent a given case study (design of a platform's mooring system for example). Each Scenario has a Layout which specifies the physical representation and connectivity of its elements such as risers, mooring lines, buoys, platforms, etc. The study is conducted by running different simulation cases on a selected Scenario, with different Analysis Parameters (numerical method, etc).

Each Simulation Case selects a combination of different environmental conditions (waves, winds and currents), prescribed motions and external forces applied in a Scenario. The output results of interest are configured in the Scenario element called OutputConfiguration. Those information are stored as DataItem elements in the Output element for each executed Simulation Case.

Although not presented here, GXML has a formal specification in the sense that for every data element present in the Domain Model there is a formal text description and some restriction rules applied to its contents, such as: cardinality; discrete or enumerated possible values; max, min and default values; pattern expressions for strings; etc. Therefore this specification, besides representing the GXML Domain Model, was also used for the creation of GXML XML Schema Definition (described in Section 4.3).

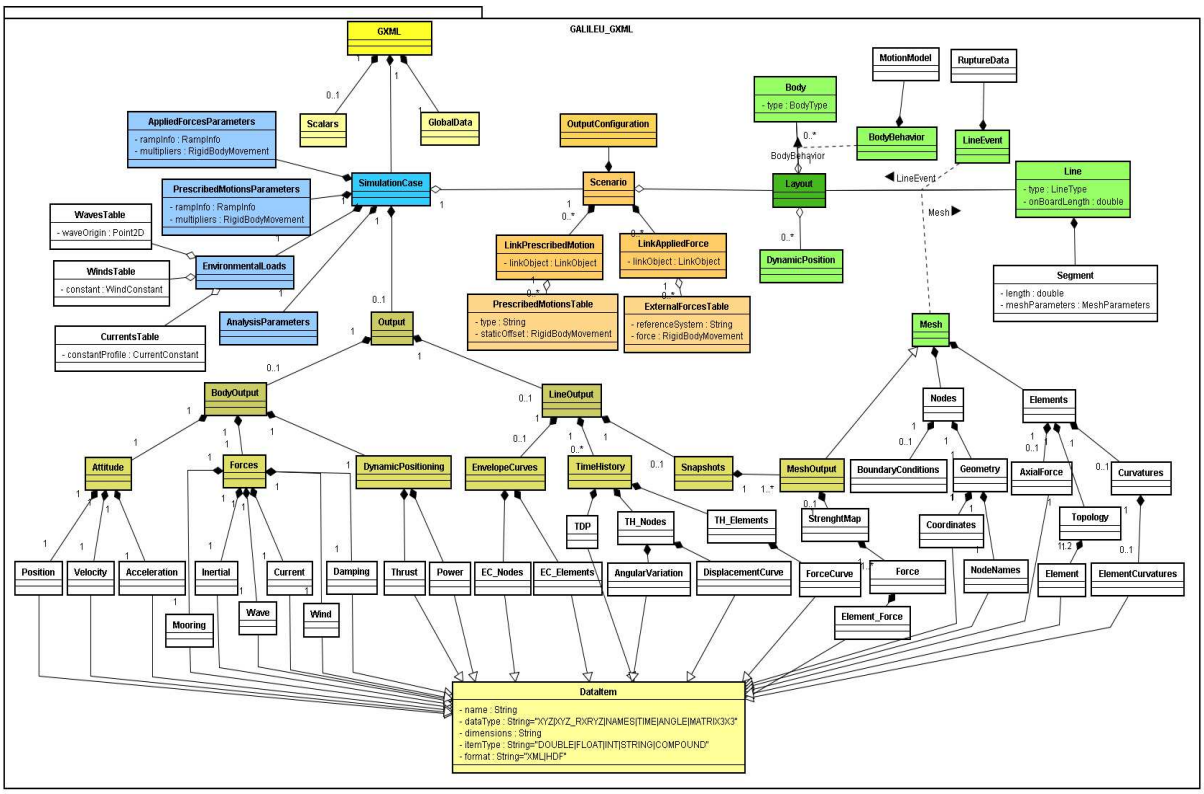

Figure 5 – Simplified GXML Domain model

4.2 Second Stage: Assembly and Verification of Files in the XDMF Format

The conclusion of the first preliminary stage of specification of contents and data structure produced the initial version of the GXML Domain Model. The teams then began to perform different tasks, aiming at the assembly and validation of the GXML Domain Model using the XDMF format. The goal was to identify which XDMF resource would be interesting to use in GXML format. Several examples of such files were manually assembled, with and without the inclusion of binary data in HDF5 files, in order to verify the adequacy of those standards to the specified data. These examples included applications of offshore systems considered in actual design activities at CENPES/Petrobras.

It is important to recall that the use of HDF5 is useful to represent large amounts of data that are typically found not only in files with results of the simulations, but also in some sections of the input files, such as those that contains the finite element mesh that represent the lines.

Figure 6 presents part of one of the XDMF files (pointing to binary data in HDF5 files) that were manually created to represent some of the GXML elements (such as meshes of the mooring lines) and to validate the structure and contents of the GXML Domain Model. This file corresponds to a model of the mooring lines and risers of the P18 semisubmersible platform. Those files could then be read by the PARAVIEW program for visualization of scientific data (Paraview 2009), which is able to read input files in the XDMF format. Figure 7 presents the PARAVIEW visualization of the P18 model described in the XDMF file of Figure 6.

```
<?xml version="1.0" ?>
<!DOCTYPE Xdmf SYSTEM "Xdmf.dtd" [ <!ENTITY HeavyData "P18_projeto.h5"> ]>
<Xdmf>
  <Domain Name="Linhas">
    <Grid Name="LINHA 1">
      <Geometry Type="XYZ">
        <DataItem ItemType="Float" Precision="8" Dimensions="131 3" Format="HDF">
          P18_projeto.h5:/Linhas/Geometry/Linha01
        </DataItem >
      </Geometry>
      <Topology ItemType="Polyline" Dimensions="130" NodesPerElement="2">
        <DataItem DataType="Int" Dimensions="130 2" Format="HDF">
          P18_projeto.h5:/Linhas/Topology/Linha01
        </DataItem>
      </Topology>
    </Grid>
     ... TRUNCATED LINES ...
    <Grid Name="ANCORAGEM 8">
      <Geometry Type="XYZ">
        <DataItem ItemType="Float" Precision="8" Dimensions="343 3" Format="HDF">
          P18_projeto.h5:/Linhas/Geometry/Linha56
         </DataItem>
      </Geometry>
      <Topology Type="Polyline" Dimensions="342" NodesPerElement="2">
        <DataItem ItemType="Int" Dimensions="342 2" Format="HDF">
          P18_projeto.h5:/Linhas/Topology/Linha56
        </DataItem>
      </Topology>
    </Grid>
  </Domain>
</Xdmf>
```

Figure 6 – Sections of the XDMF-HDF5, Mesh of the Lines of the P18 Platform

Regarding the full use of the XDMF standard, it is important to recall that it is still undergoing new developments and changes; therefore it does not yet have a stable specification (Clarke & Mark, 2007; Elias et al., 2009). Moreover, it does not express all types of data that are incorporated in the specification of the unified Galileo data format described previously (the GXML Domain Model), such as environmental load conditions, among others. Another important aspect is the semantics issue. XDMF is a very good format to represent basic meshes, but the Galileo project needs more data types. The *Information* element of XDMF could have been used, but semantics would be lost, which is very important when interoperability is considered.

Nonetheless the XDMF conceptual design has inspired GXML design tremendously. The XDMF concept of Grid element was extended in GXML as a Mesh element not only reusing Geometry and Topology elements but also including further information to allow different Galileo's applications to consistently and uniformly build the same final mesh for the same Line element (mooring line or a riser). The DataItem element was replicated in GXML, with some minor changes, once it was considered an appropriate abstraction for dealing with elements that can be stored in HDF files (heavy data) or inside the GXML file (light data).

During the development of the GXML, as described in the following sections, the specification is proceeding as a "super-set" of the current XDMF format, but without necessarily completely adhering to it. Alternatively, in the future, the Galileo team could try

to interact with the development team of the XDMF standard, offering support and suggesting adaptations to conform to the needs of the Galileo specifications. This way GXML could become a format associated to XDMF, moving towards an offshore engineering simulation standard perhaps.

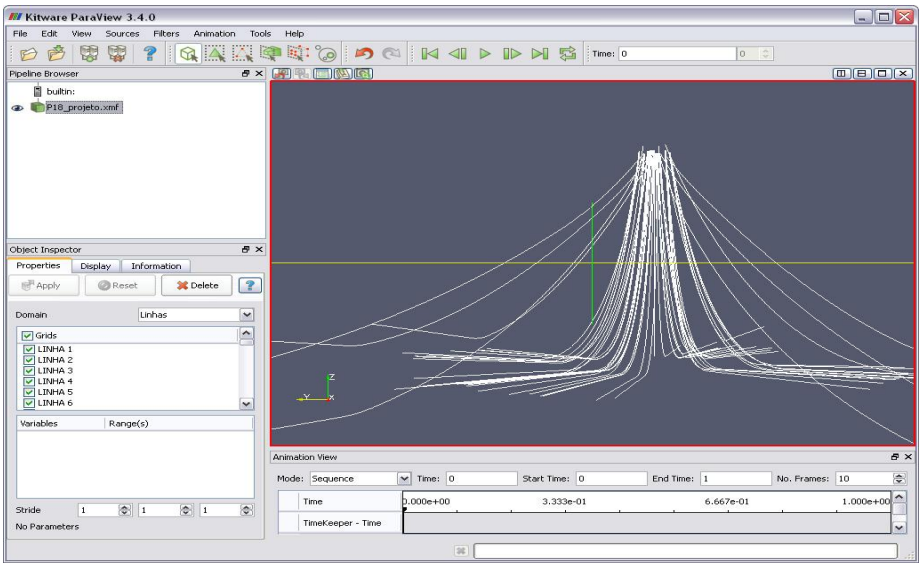

Figure 7 - The main Paraview screen: visualization of the lines of P18 platform

4.3 Third Stage: New Spiral for Formal Specification – XSD files

Subsequently, a new spiral of activities related to the specification of contents and structure of the data to be included in the new format was performed by the whole team, aiming at the continuous evolution of the GXML Domain Model.

At this stage, the teams worked directly over the preliminary XML specifications that resulted from the activities of the previous stages. Therefore, the specification was incrementally improved as long as the representation of all the Domain Model elements was evaluated. Several real examples were built using different scenarios with its corresponding layouts (including hulls and lines). Many simulation cases including environmental loadings, external forces, prescribed motions, static and dynamic analysis parameters, and its associated output responses were also reproduced using GXML format specification.

It is worthwhile to mention that the large amount of data, and the complexity of their relationships, has turned the refinement of the GXML Domain Model into a lengthy incremental and iterative process. That is, after each meeting, a new concept was defined, and the previous concepts were revised against the new ones, leading to frequent modifications in the representation of the information, until a consistent and efficient specification structure was reached. As an example, on the preliminary versions of the specification, the analysis parameters for hulls and lines were dispersed inside the respective groups of data; on a more recent version of the specification, they were all grouped in a single specific set.

As the result of these efforts, a reference document was generated, containing the specification of the elements of the GXML Domain Model and its associated format. This specification represents the relationship between the elements, through the definition of groups and subgroups; indicates which elements are mandatory or optional; and includes their description in terms of element type, its cardinality, meaning and functionality, as well as the indication of default values and restrictions on its values (min, max) or enumerated discrete values and patterns for strings governed by regular expressions, when applicable.

From this reference document, a formal specification of the GXML file was generated, established in terms of XSD (XML *Schema Definition*) files (Fallside & Walmsley, 2004). XML Schema can be connected to XML documents to restrict their content. It defines the set of elements allowed in the XML document along with the data types of each element. XML Schema has pre-defined types such as Integer, String, among others, that can be assigned to textual elements (elements in which the content is non-structured). Defining such restrictions is important in interoperability scenarios, since the applications "know" the schema and, as a consequence, know what to expect from the XML document that conforms to that schema. In fact, XML APIs provide a validation operation, that verifies whether a given XML document satisfies the restrictions imposed by its schema or not.

Since the XML Schema specification is verbose, sections of the current GXML schema are presented in Figure 8. In this figure, XML elements have "< >" in front of their names, while attributes have "@". Optional elements are represented with the "< ? >" symbol. Figure 9 presents part of the XSD declaration of the GXML Global Data element.

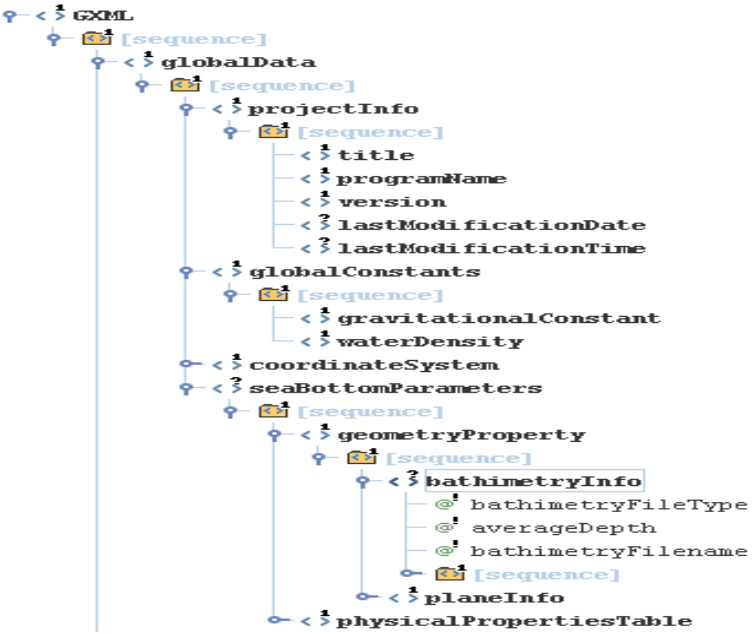

Figure 8 - Sections of the XSD specification for floating bodies

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema targetNamespace="GALILEU" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="GALILEU" elementFormDefault="qualified">
... TRUNCATED LINES ...
<xs:complexType name="tGlobalData">
  <xs:sequence>
    <xs:element name="projectInfo" type="tProjectInfo">
      <xs:annotation>
        <xs:documentation source="comment">Informação do projeto</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="globalConstants" type="tGlobalDataConstants">
      <xs:annotation>
        <xs:documentation source="comment">Constantes globais</xs:documentation>
      </xs:annotation>
    </xs:element>
    ... TRUNCATED LINES ...
</xs:complexType>
... TRUNCATED LINES ...
</xs:schema>
```

Figure 9 - Section of the GXML-XSD for Global Data

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="GXMLToHtml.xslt"?>
<GXML xmlns="GALILEU" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" … >
    <globalData>
        <projectInfo>
            <title>MODELO DA P10 COM ANCORAGEM; LDA=1200 m V2.5e</title>
            <programName>SITUA - COPPE/UFRJ</programName>
            <version>1.8.18</version>
            <lastModificationDate>2005-03-29</lastModificationDate>
            <lastModificationTime>13:24:36.000</lastModificationTime>
        </projectInfo>
        <globalConstants>
            <gravitationalConstant>9.806</gravitationalConstant>
            <waterDensity>1.025</waterDensity>
        </globalConstants>
        ... TRUNCATED LINES ...
</GXML>
```

Figure 10 - Section of the GXML for the P10 Platform

Finally, based on the GXML-XSD specification, two XSLT Transformations (Clark, 1999) were created to allow a better visualization of GXML files, one for converting GXML files to HTML and another to convert GXML to XDMF. XSLT is a transformation language for XML, able of transforming XML Documents into any other text-based format. Figure 10 presents part of GXML file representing a model of the P10 platform, while Figure 11 presents the same GXML file in a web browser (transformed to HTML using XSLT).

Since there is more available information to be visualized from a GXML file than what could be translated into XDMF, the Galileo team has decided to build a GXML specialized plugin for Environ, a VR visualization tool specialized for large Engineering CAD models (Santos et al., 2008a). Figure 12 shows a snapshot of the simulation results for the P18 platform in Environ.

Figure 11 – Visualization of GXML data of P18 platform in a web browser

It is important to emphasize the benefits of having XDMF (and HDF5) as a subset of the GXML definition, which allows a rapid visualization of the results of a particular simulation of any Galileo's application in PARAVIEW through the use of a GXML XSLT transformation. If a more sophisticated visualization is necessary, such as in a large display or in an immersive virtual environment, the Environ plugin can be used accordingly.

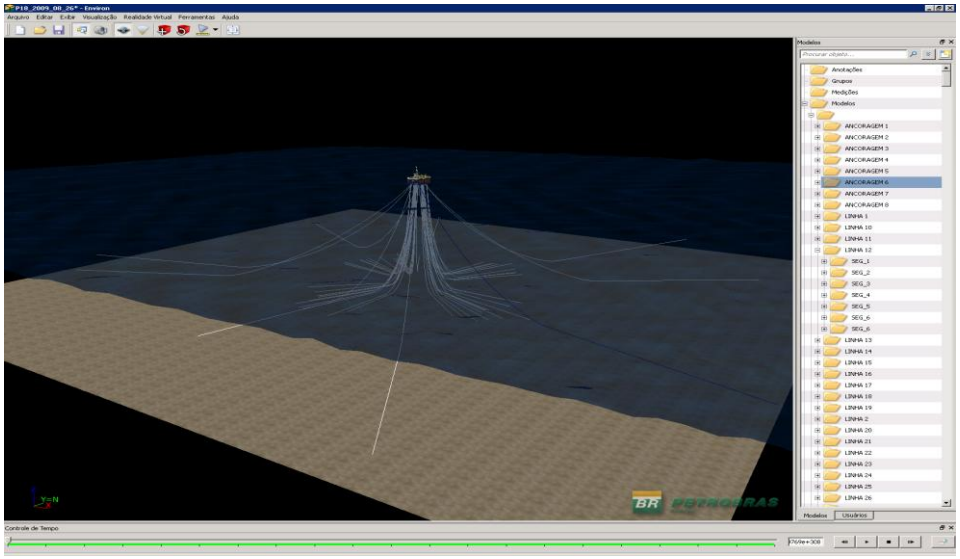

Figure 12 – Visualization of GXML data for P18 plaform

5 SUMMARY OF GXML ARCHITECTURE

Due to the Galileo's interoperability requirements, GXML was devised not only as a format for data sharing but also as a software abstraction layer, with its corresponding application program interface (GXML API), that isolates Galileo's applications from the need of criticizing input data and also dealing with I/O operations and its dependencies.

Thus, the main goal of GXML is twofold: to hide the complexity of handling XML files from the host applications; and also to provide efficient ways for reading and writing huge volume of data through the use of the Parallel HDF5 library, a high performance parallel I/O API for message passing environment (HDF Group, 2008).

To hide the complexity of parsing XML documents from the host applications, an XML Data Binding was used. XML Data Binding refers to the process of representing the information of a XML document as objects in the computer's memory. This allows applications to access XML data directly from those objects rather than using the DOM or SAX API (which are standard APIs to access XML data) to retrieve the data from a direct representation of the XML itself. Ultimately, it consists in providing a set of classes that represents the XML elements and their content. Thus, applications use objects from these classes as input, instead of dealing with XML elements in main memory. In our case, a C++ XML binding open-source library (CodeSynthesis, 2009) is being used to generate GXML binding classes.

Most of the applications of the Galileo project were written in Fortran or C programming languages. Thus, binding to those languages must be provided, allowing them to access the C++ classes of the GXML API, the Façade layer of Figure 13. Lua and Phython bindings are also provided allowing scripting the GXML into dynamic applications. The GXML library is thus composed of the following components (represented in Figure 13):

(1) High level GXML API, provided by the GXML::Facade which manipulates GXML::Core objects, the dynamic GXML::Proxy objects, which ultimately represents the Domain Model objects for the host application, and the GXML::DataItem objects which encapsulates the access to the HDF5 files;
(2) Reflection C++ API for generating the dynamic GXML::Proxy objects;
(3) XML Data Binding layer provided by the CodeSynthesis-XSD library;
(4) HDF5 and Parallel HDF5 libraries for manipulating HDF5 files.

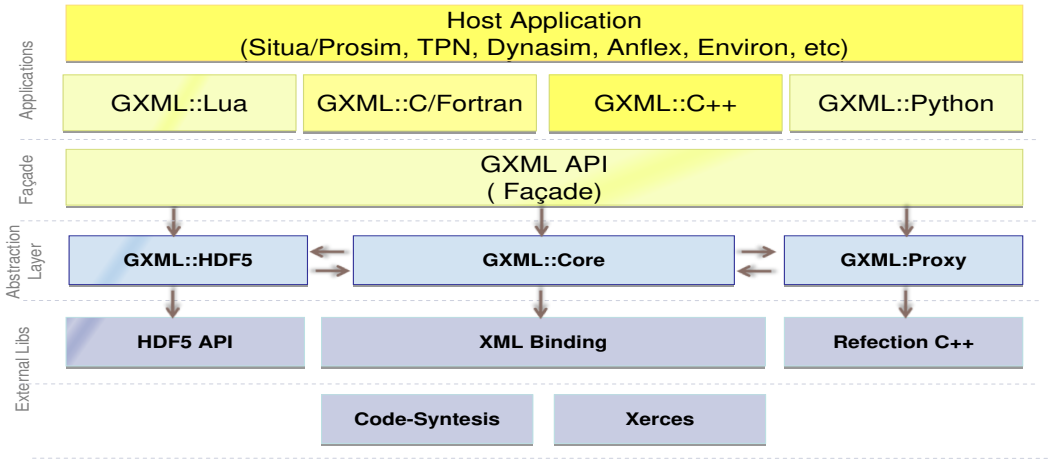

Figure 13 - GXML Lib and API

It is important to notice that in this architecture the use of a particular implementation of a XML Data Binding provides an undesirable dependency. In order to promote the decoupling of the GXML API from any particular XML binding implementation, a C++ Reflection API is being used to create Dynamic Proxies for every generated XML binding class.

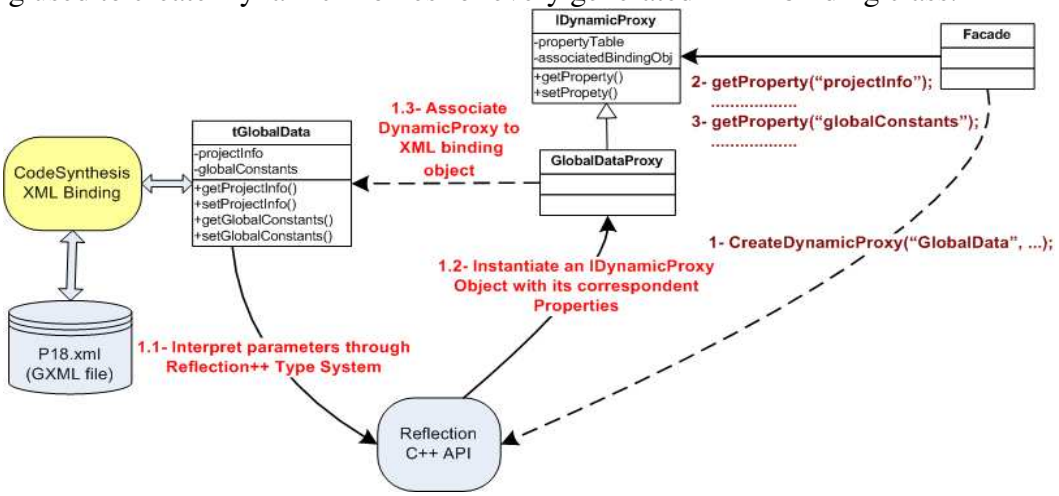

Figure 14 – Dynamic Proxy of XML Binding objects generation

Figure 14 presents a sequence of operations that the Facade object should perform in order to fulfill host application requests for reading or writing the contents of GlobalData GXML elements (projectInfo, globalConstants, etc). In the first step the Facade object send a request for the creation of a dynamic proxy (in this case GlobalDataProxy) to the Reflection C++ API (Figure 14, message 1). Through the use of reflection methods over the XML binding object created by the Code Synthesis (in this example, tGlobalData), an instance of a dynamic proxy is tailored upon the information obtained by reflection (Figure 14, messages 1.1, 1.2 and 1.3). After receiving a reference to the instance of the proxy object, specially tailored for him, the Facade object can now access the content of the proxy object through the methods available in the IDynamicProxy interface (Figure 14, messages 2 and 3).

Figure 15 shows a usage scenario of this architecture in a hypothetic situation of a host application reading a GXML file that could have data of a P18 platform and lots of mooring lines for starting a specific simulation. Upon parsing the GXML file, a GXML tree of DynamicProxy objects is constructed in memory. This tree is parallel to the tree created by the XML binding implementation used, in our case the CodeSysnthesis library. It is important

to emphasize that there is no duplication of final data in those two trees once the information is actually stored in the XML binding tree. During its construction through the Reflection API, every node of the GXML tree has a map of properties where each property has a name (identical to element name in the xml file), and a reference to the final data (the leaf nodes) of the XML binding tree. Besides that, the HDF5 are read only when the host application needs it. Thus, the GXML memory requirements are not considered an issue, even for very large simulations as the ones that were tested until now.

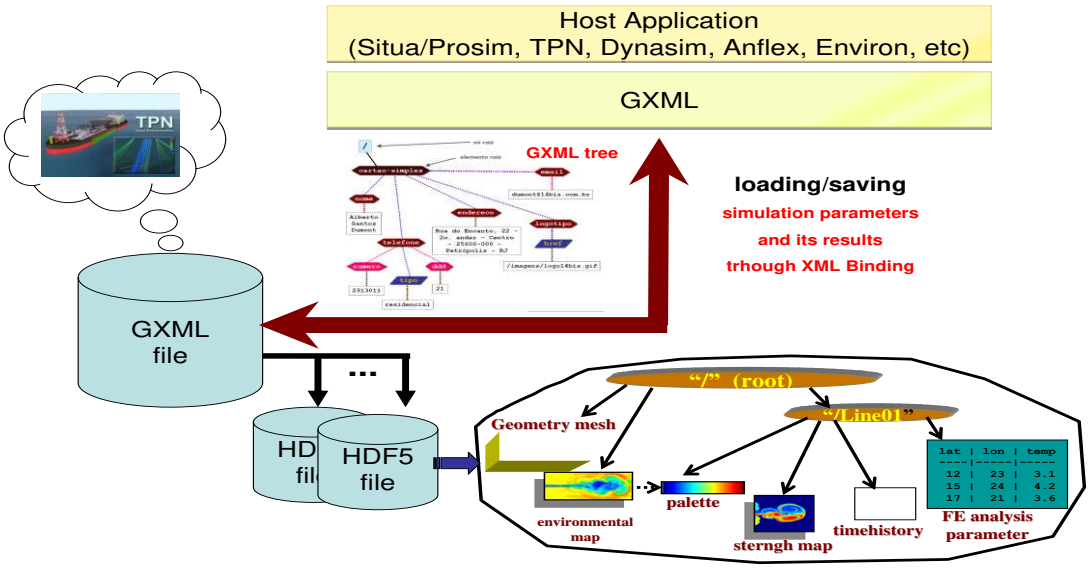

Figure 15 - GXML Usage Scenario

6 FINAL REMARKS

This paper presented the GXML format, an XML-based format for data files aimed at solving the interoperability problems of the Galileo applications, together with the architecture that provides transparent access of XML data to the Galileo's applications.

The use of XML in the context of the Galileo project has raised some concerns: (1) Scalability when dealing with large volumes of data; (2) Seamlessly integration into host applications; (3) Creation of an abstraction layer to isolate host applications from XML file access.

Scalability is addressed because GXML has been defined as a superset of XDMF (Clarke & Mark, 2007; Elias et al., 2009), an XML format that is able to handle light data and to describe heavy data that is actually stored as HDF5 files.

The second and third concerns are addressed by the GXML API (Application Program Interface). Some changes, however, will have to be made to the current applications. Today, most of the Galileo applications use text-based files, and parses them to input data. Thus, they will have only to modify the parsing methods and replace them by calls to the API that was built. This will largely improve interoperability between Galileo applications, since no intermediate conversion steps will be needed when the project is complete.

This interoperable capability will open new possibilities for orchestrating the applications in different sequences, comprising specific scientific workflows specially tailored for solving even more complex problems (Santos et al., 2008a, and Carvalho, 2009). Another important aspect that will be investigated is the use of the parallel HDF5 library to speed up the execution of the Galileo's applications.

It is expected that, once relieved from the responsibility of implementing efficient and flexible I/O operations, the engineers will be able to concentrate on the development of more powerful applications towards a Petascale High Performance Computational level which is one of the main objectives of the Galileo project.

Acknowledgements. The authors would like to thank CNPq and Petrobras for financial support.

REFERENCES

Blackburn, K., Lazzarini, A., Prince, T., and Williams, R.,1999. *XSIL: Extensible Scientific Interchange Language.* Technical Report Caltech. [CaltechCACR: CACR-1999-171]

Caron, J., Cinquini, L., Davis, E., Drach, B., Nativi, S., Rew, R., 2008. *The NetCFD Markup Language*. http://www.unidata.ucar.edu/software/netcdf/ncml/. Last visited Sept 2009.

Carvalho L.O.M., Corrêa F.N., Mattoso M., Jacob B.P., 2009, *Application of Scientific Workflows to the Design of Risers for Offshore Oil Production* (in Portuguese). In: Procs. of CILAMCE-2009 – XXX Iberian-Latin-American Congress on Computational Methods in Engineering, Búzios, RJ, Brasil.

Clark, J., 1999. XSL Transformations (XSLT). W3C Recommendation 16 November 1999. Available at http://www.w3.org/TR/xslt. Last visited on September 2009.

Clarke, J., Mark, E., 2007. *Enhancements to the eXtensible Data Model and Format (XDMF)*. In: DoD High Perf Comp Modern. Program UGroup Conference (HPCMP), pp. 322-327.

CodeSynthesis, 2009. Code Synthesis - XML Data Binding for C++. Website, Available at http://www.codesynthesis.com/products/xsd/. Last visited on September 2009.

Deelman, E., Gannon, D., Shields, M., Taylor, I., 2009. Workflows and e-Science: An overview of workflow system features and capabilities, *Future Generation Computer Systems*, v25,p.528-540.

Elias, R., Braganholo, V., Clarke, J., Mattoso, M., Coutinho, A., 2009. *Using XML with Large Parallel DataSets: is there any hope?* In: Parallel Computat. Fluid Dynamics(ParCFD).

Ensight, 2009. *User Manual*. Available at http://www.ensight.com. Last visited on Sept 2009.

Fallside, D., Walmsley, P., 2004. XML Schema Part 0: Primer Second Ed. W3C Rec 28 Oct 2004, Available at: http://www.w3.org/XML/Schema. Last visited on September 2009.

Hartnett, E., Rew, R., 2008. *Experience with an enhanced netCDF data model and interface for scientific data access*. In: AMS IIPS.

HDF Group, 2008. *HDF5 File Format Specification Version 2.0*. Available at http://www.hdfgroup.org/HDF5/doc/H5.format.html. Last visited on September, 2009.

Jacob, B., 2006. PROSIM Program: Numerical Simulation of the Behavior of Offshore Oil Exploitation Systems, Theoretical Manual (in Portuguese), LAMCSO/COPPE/UFRJ, RJ.

Moreau, L., Zhao, Y., Foster, I., Voeckler, J., Wilde, M., 2005. *XDTM: The XML Data Type and Mapping for Specifying Datasets*. In: European Grid Conference (EGC), Lecture Notes in Computer Science 3470, 495-505.

Oinn, T., Addis, M., Ferris, J., Marvin, D., Greenwood, M., Goble, C., Wipat, A., Li, P., Carver, T., 2004. *Delivering web service coordination capability to users*. In: International World Wide Web Conference on Alternate papers & posters, p. 438-439.

Paraview, 2009. Available website http://www.paraview.org. Last visited on September 2009.

Santos, I. H. F., Raposo, A. B., Soares, L. P., Corseuil, E. T. L., Wagner, G. N., Santos, P. I. N., Toledo, R. Gattass, M. *EnViron: An Integrated VR Tool for Engineering Projects*. 12th International Conference on CSCW in Design, CSCWD08. Xi'an,China,Abril 2008.

Santos, I. H. F., Raposo, A. B., Gattass, M. *A Software Architecture for an Engineering Collaborative Problem Solving Environment*. In:IEEE Soft. Eng. Work, SEW 2008, p.43-51. Kasandra, Greece.

Soares, L. P.; Corseuil, E. T. L.; Raposo, A. B.; Gattass, M.; Santos, I. H. F. *Environ: Uma Ferramenta de Realidade Virtual para Projetos de Engenharia*. CILAMCE-Iberian Latin American Congress on Comp Methods in Engineering, 2008, 17p. Maceió, AL, Brasil.

Schroeder,W., Martin,K., Lorensen, B., 2006. *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, 4th Edition.