POOL DEVELOPMENTS FOR OBJECT PERSISTENCY INTO RELATIONAL DATABASES

G. Govi, R. Chytracek, D. Düllmann, I. Papadopoulos (CERN, 1211 Geneve 23, Switzerland)
Z. Xie (Princeton University, Princeton, NJ 08544, USA)

Abstract

The LCG POOL project has recently moved his focus on the developments of storage back-ends based on Relational Databases. Following the requirements of the LHC experiments, POOL has developed a software package for the object persistency into Relational Tables. This paper will describe the main functionality of the package, explaining how the mechanism provided by POOL allows efficiently storing and retrieving arbitrary user objects.

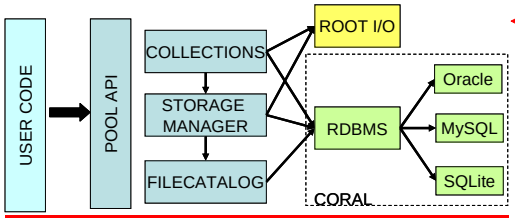


Fig. 1. Decomposition of the POOL API in three domains.

INTRODUCTION

The LCG [1] POOL [2][3] project provides a software framework for the persistency of the LHC experiment data.

The main strategy of the project has been to satisfy the requirement to provide access to a variety of persistency technologies, allowing for possible changes during the LHC lifetime.

The software developed in the first phase of the project, integrates seamless a streaming technology (eg ROOT I/O [4]) for complex object storage [5]. During the last year the POOL persistency framework has been adopted by three LHC experiment (ATLAS, CMS and LHCb), integrated into their offline software and used in large-scale production activities. The POOL API has been fully validated and it has been demonstrated to meet most of the requirements for production.

More recently RDBMS technology has been introduced for consistent metadata handling with transactional access. In order to provide support for more relational database systems, the low-level code handling the hand-shake with the database has been factorized in a generic API (COmmon Relational Abstraction Layer) with a specific implementation for each DB vendor supported. In this way, all the database-specific SQL needed to operate on the different RDBMS is hidden by dedicated plug-ins, and all the POOL components access different RDBMS technologies through a single, uniform protocol.

In particular, the extension of the RDBMS backend to the POOL Storage Manager component, allows addressing use cases of C++ object persistency not yet satisfied by the previous POOL components, such as the handling and management of Condition data and Detector online data.

MOTIVATION

The need of a relational back-end for the C++ object persistency has been expressed by the experiments since the early definition of the POOL project mandate. In this context, two main physics use cases have been identified, mainly related to the storage of condition, configuration and detector data.

The first use case concerns the storage of detector-related data. Typically, the experiment data models for event data involve complex hierarchal structures, described by more non-trivial object types. C++ objects of this type are conveniently stored in files, using ROOT I/O streaming mechanism. Conversely, a large category of detector-related data, like general condition measurements, configuration and calibration values, and geometry data, are described by simpler structures, often associated to an *Interval of Validity* and other metadata attributes. The handling of this type of data frequently involves selection applied either on the payload data or on their associated metadata, which are hardly supported by persistency mechanism based on streaming to files.

For this reason, it appears appropriate to select a storage system capable to easily reflect the granularity of the data model, preserving explicitly the relationships between the various entities. Relational Databases are well suited for such requirements.

A second use case arises from the fact that configuration and detector control data are written by online processes directly to relational databases using native APIs or vendor-specific tools. Off-line reconstruction and analysis frameworks often require such data to be read in as software objects, which can be referenced by other reconstruction or analysis objects. An example would be a reconstructed event header pointing to objects holding information such as the beam luminosity or the detector layout corresponding to the

Formatted: No bullets or numbering
Deleted: PREPARATION OF PAPER CONTRIBUTIONS FOR PUBLICATION IN THE CHEP04 PROCEEDINGS*
Formatted: English (U.S.)
Formatted: Author List, Justified
Formatted: English (U.S.)
Formatted: French (France)
Deleted: J. Poole, C. Petit-Jean-Genaz, CERN, Geneva, Switzerland
Formatted: Body Text Indent, Centered
Deleted:
Formatted: English (U.S.)
Formatted: Font: 12 pt, Italian (Italy)
Deleted: P. Lucas#, FNAL, Batavia, IL 60510, USA¶
Formatted: English (U.S.)
Formatted: English (U.S.), Kern at 8 pt
Formatted: English (U.S.)
Formatted: Centered
Formatted: Body Text Indent
Formatted: Kern at 8 pt
Formatted: Font: 8 pt, Not Bold
Formatted: Caption, Indent: Left: 0.14"
Deleted: This document describes the requirements and the format for the submission of papers to the CHEP 2004
Formatted: No bullets or numbering
Deleted:
Deleted: conference. It has been adapted from the template that can be found on the Joint Accelerator Conference Website (JACoW)[1] that is used for the publication of the proceedings of several accelerator conferences (APAC, EPAC, PAC). It is not intended that this should be a tutorial in word processing; the aim is only ... [1]
Deleted: ¶
Deleted: ¶
Deleted:
Deleted: This document also pr ... [2]
Deleted: UBMISSION OF PAPERS
Formatted: Font: Italic
Formatted: Font: Italic
Formatted ... [3]

time that the actual physics event took place. A relational back-end for the POOL Storage Manager would have to handle existing relational data which have to be presented as user-defined software objects.

ARCHITECTURE

The POOL relational back-end comprises three main domains, as shown in Fig.2:

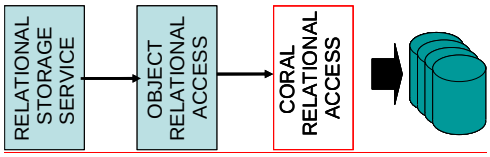


Fig. 2. The components comprising the POOL relational back-end.

- The Common Relational Abstraction Layer (CORAL), which defines a technologically neutral API for accessing and manipulating data and schemas in relational databases.
- The Object-Relational Access mapping mechanism, which is responsible for transforming C++ object definitions to relational structures and vice-versa.
- The Relational Storage Service, which is an adapter implementing the POOL Storage Service interfaces in terms of the CORAL and using the Object-Relational Access mapping mechanism.

The Common Relational Abstraction Layer

The CORAL [7] has been identified as the base domain for the whole relational back-end for several reasons: it is required in order to achieve vendor independence; it is useful in developing the relational components of POOL (File Catalogue, and Collections) and the ConditionsDB; and it is potentially useful in enabling user application access to relational data. Moreover, its introduction may address the problem of distributing data in RDBMS' of different flavors.

The CORAL abstract interfaces are defined in the *RelationalAccess* package. Their technology-specific realizations are implemented following the SEAL component model [8] as plug-in libraries. This architecture reduces the code maintenance effort for the relational components and allows efficient bug tracing. In providing access to specific RDBMS technologies via plug-ins, the risk of binding to a particular RDBMS vendor is minimized. Furthermore it allows the usage of multiple technologies in parallel. Applications which access relational databases through the CORAL automatically become testing grounds for plug-ins of new RDBMS flavors.

OBJECT STORAGE MECHANISM

The second domain in the POOL relational back-end addresses the issues which emerge when a C++ class is to be mapped to a relational structure, and vice-versa.

In the relational world tables are broadly equivalent to classes in the object world: they define how data are laid out in memory. Rows in a table can be thought of as the equivalent of objects of a class because they hold data of a well defined layout.

The first fundamental difference between objects and rows is that the former exhibit identity by construction while the latter by default not. Identity is necessary to uniquely and unambiguously address an object in a program in order to access its data. It is also the basis of every association between objects. To solve the problem of missing identity it is required that rows which are to be represented as objects should be in tables which define a primary key or a unique index.

The second difference between objects and rows derives from the associations between two or more data sets. In the object world there are aggregations (associations realized as persistent references) and compositions. In the relational world the corresponding constructs are foreign key constraints.

Object associations have a well defined directionality and multiplicity. On the other hand a table schema alone cannot determine unambiguously the directionality and the multiplicity implied by a foreign key constraint. It is up to the mapping process to resolve these ambiguities.

To illustrate how the mapping works let us assume that a user would like to store objects of a simple C++ class which contains two simple members, and more complex vector member:

```
class A {
    int i;
    float x;
    class B {
     float x;
     float y;
    };
    std::vector<B> b;
};
```

One of the possible mappings to a relational schema for this class is presented in Fig.3. The schema contains one table (T_A) for the *top-level* class A, and another one (T_B) to accommodate the values of the data member vector m_b. The primary key (ID) in T_A serves the role of the object identity. In the table T_B a foreign key constraint is defined. There is also a special column to hold the position of the elements inside the vector.

Formatted: Font color: Auto, English (U.K.)

Formatted: Heading 2,Section Heading, Space Before: 0 pt, After: 0 pt, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Formatted: Indent: First line: 0"

Formatted: Font: Courier

Formatted: Font: Courier

Formatted: Font: Courier

Formatted: Font: Courier

Formatted: Font: Courier

Formatted: Font: Courier

Formatted: Font: Courier

Formatted: Font: Courier

Formatted: Font: Courier

Formatted: Font: Courier

Formatted: Font: Italic

P.K. T_A F.K. T_B

ID	M_I	M_X
1	14	0.3
2	34	4.6
⋮	⋮	⋮

ID	POS	M_X	M_Y
1	1	0.1	2.3
1	2	0.2	4.5
1	3	0.4	6.4
2	1	1.4	6.6
2	2	0.3	1.3
2	3	0.1	3.2
⋮	⋮	⋮	⋮

Fig. 3. The relational schema corresponding to a mapped class.

An alternative way to store set of objects in arrays as a whole is to stream the elements in a single binary entity, which is stored a *Binary Large Object* (BLOB) into the relational database.

The *ObjectRelationalAccess* package of POOL provides the software for generating mappings for a given a class. It allows a user to prepare the relational schema by creating or altering the relevant tables. While there are default rules for generating mappings, a user can override them. This would be the case if, for example, one would like to generate object-relational mappings for existing data. POOL provides a tool which uses an XML file to steer mapping generation; the non-default rules are specified using an XML schema. The storage method for the arrays can be also selectively defined among the C++ types involved in the application, using the XML driver file.

The generated mapping is a hierarchical structure of elements describing the C++ types and names of the data members as well as the names of the associated columns and tables. The mapping hierarchy is versioned and can be stored in the database in three *hidden* tables.

Object storage and retrieval is performed using ROOT Reflection [9] information for the C++ class of interest, and the corresponding mapping element for this class. The version of the mapping ensures that simple schema evolution cases are handled automatically.

A POOL *container* of objects simply records primary key values, and the mapping versions corresponding to an object whose data members are written to the relational tables. The POOL *RelationalStorageService* component, which will be released this year, will ensure that full object I/O can be performed through the POOL framework in an identical -to the user- way with the existing object streaming to ROOT files.

OBJECT-BASED READING OF PRE-EXISTING TABLES

As mentioned above, a main requirement of the POOL Relational Storage Service is the capability to read data stored in pre-existing relational tables as C++ object of user-defined classes.

As explained in the previous paragraph, the mapping between the user C++ classes and the relational entities (tables and columns) can be defined according to the user needs, with the help of an XML driver file. In this way, each class attribute can be associated to a column of the existing tables.

The POOL storage also requires a few additional tables, which are necessary to store the header data for the POOL containers and other special metadata. To construct these tables and fill them with the necessary data, the POOL API delivers a dedicated command line tool.

The tool requires as input an XML driver file where the user defines the POOL Containers to be associated to the C++ classes involved, for which a mapping to the relational tables has been previously defined (Fig.4).

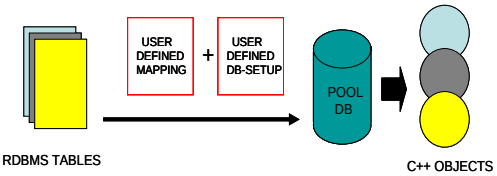


Fig. 4. The creation of a POOL database from existing Relational tables.

SUMMARY

The LCG POOL project provides a software framework for the persistency of the LHC experiment data.

A new relational back-end for the POOL Storage Manager has been implemented based on the CORAL package, with the double purpose of storing data described as user-defined objects in relational table, or conversely presenting data already stored in relational table as objects. These functionalities are addressing two use cases not yet satisfied by the previous POOL components, such as the handling and management of Condition data and Detector online data.

The future POOL developments will be focused on tuning and improving the performance of data access, reducing the penalty introduced by the additional layers of the framework – CORAL in particular – and optimizing the internal procedures for the data presentation as objects. Furthermore, the requests of the LHC experiments, necessitating changes in the existing code or implementation of new features, will be followed up in the POOL project plans.

Formatted: Keep with next

Formatted: Indent: First line: 0.13", Keep with next

Formatted: English (U.K.)

Formatted: Indent: First line: 0"

Formatted: Keep with next

Formatted: Font: 8 pt

Formatted: Normal, Indent: Left: 0.14", Don't adjust space between Latin and Asian text, Don't adjust space between Asian text and numbers

Formatted: Font: 8 pt

Formatted: Font: 8 pt

Formatted: Keep with next

Deleted: Each author should submit from the CHEP04 website one single PDF file and all source files of the paper contribution.¶

Formatted: Normal, Indent: First line: 0.14", Don't adjust space between Latin and Asian text, Don't adjust space between Asian text and numbers

Formatted: Font: Times New Roman, Font color: Black, English (U.S.)

Deleted: <sp>

Formatted: Indent: First line: 0"

REFERENCES

[1] The LHC Computing Grid
http://lcg.web.cern.ch
[2] The POOL Project,
http://pool.cern.ch
[3] D. Duellmann,
"The LCG POOL Project General Overview and Project Structure", CHEP 2003 Proceedings, MOKT007
[4] R.Brun and F.Rademakers, "ROOT-An Object Oriented Data Analysis Framework",
Nucl. Inst.& Meth. in Phys.Res.A389(1997)81-86.
see also: http://root.cern.ch
[5] M.Frank *et al.*, "The POOL Data Storage, Cache and Conversion Mechanism".
CHEP 2003, proceeding, MOKT008
[6] Z.Xie *at al.*, "POOL File Catalog, Collection and Meta Data Components",
CHEP 2003, proceeding, MOKT009
[7] I.Papadopoulos,
"CORAL, a software system for vendor-neutral access to relational databases", Contribution #329, this conference.
[8] Radovan Chytracek et al.,
"The SEAL Component Model", CHEP 2004 Proceedings.
[9] Stefan Roiser et al.,
"Reflex, reflection for C++", Contribution #185, this conference.

Deleted: Microsoft

Deleted:

Deleted: Word

Deleted: help files. ¶
If a displayed equation needs a number, place it flush with the right margin of the column (see Eq. 1). Units should b ... [4]

Deleted: and

Deleted: Microsoft Word and ... [5]

Deleted: k

Deleted: processor please respe ... [6]

Formatted: French (France)

Deleted: ¶

Deleted: the PDF file of the pap ... [7]

Formatted ... [8]

Deleted: Templates are provided for

Deleted: Microsoft Word and

Deleted: LaTeX, Microsoft Wor ... [9]

Deleted: <#>postscript

Deleted: <#>PDF file.¶ ... [10]

Deleted: [1

Deleted:

Formatted ... [11]

Formatted ... [12]

Deleted: http://www.jacow.org/.

Formatted ... [13]

Deleted: 2

Deleted:

Deleted: A. Name and D. Pers ... [14]

Deleted: 3

Deleted:

Formatted: Left

Deleted: A.N. Other, "A Very ... [15]

Formatted: Body Text Indent

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Formatted: Font: 8 pt

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Formatted: Default, Left

Formatted: Body Text Indent, Left

Formatted ... [16]

Formatted: English (U.S.)

Formatted: Body Text Indent

Deleted: Section Break (Continuous)

Formatted: Font: Times New Roman

Page 1: [1] Deleted govi 10/5/2004 3:14:00 PM

CONFERENCE. IT HAS BEEN ADAPTED FROM THE TEMPLATE THAT CAN BE FOUND ON THE JOINT ACCELERATOR CONFERENCE WEBSITE (JACOW)[1] THAT IS USED FOR THE PUBLICATION OF THE PROCEEDINGS OF SEVERAL ACCELERATOR CONFERENCES (APAC, EPAC, PAC). IT IS NOT INTENDED THAT THIS SHOULD BE A TUTORIAL IN WORD PROCESSING; THE AIM IS ONLY TO EXPLAIN THE PARTICULAR REQUIREMENTS FOR ELECTRONIC PUBLICATION.

Page 1: [2] Deleted govi 10/5/2004 3:14:00 PM

This document also provides an example of using the recommended format for paper contributions. For any questions regarding this template contact the CHEP04 Programme Committee Secretary (pc-secretary@chep04.org).

S

Page 1: [3] Formatted govi 4/21/2006 2:43:00 PM

Font: Times New Roman, Font color: Black, English (U.S.)

Page 4: [4] Deleted govi 10/8/2004 6:47:00 PM

help files.

If a displayed equation needs a number, place it flush with the right margin of the column (see Eq. 1). Units should be written using the roman font, not the italic font.

$$C_B = \frac{q^3}{3\varepsilon_0 mc} = 3.54\mu\text{eV/T} \qquad (1)$$

References

All bibliographical and web references should be numbered and listed at the end of the paper in a section called “References.” When referring to a reference in the text, place the corresponding reference number in square brackets [3].

Footnotes

Footnotes on the title and author lines may be used for acknowledgements, affiliations and e-mail addresses. A nonnumeric sequence of characters (*, #, †, ‡) should be used. All other footnotes should be included in the reference section and use the normal numeric sequencing.

Word users—do not use Word’s footnote feature (**Insert**, **Footnote**) to insert footnotes, as this will create formatting problems. Instead, insert footnotes manually in a text box at the bottom of the first column. Footnotes should only appear at the bottom of the first column on the first page.

Acronyms

Acronyms should be defined the first time they appear.

PAGE NUMBERS

DO NOT include any page numbers. The editorial staff will add them when they produce the final proceedings.

TEMPLATES

THIS DOCUMENT IS AN EXAMPLE OF USAGE OF THE TEMPLATE PROVIDED FOR AUTHORS. TEMPLATE DOCUMENTS FOR LATEX,

Page 4: [5] Deleted **govi** **10/8/2004 6:47:00 PM**

MICROSOFT WORD AND OPENOFFICE ARE AVAILABLE FROM THE CHEP 2004 WEBSITE. IF YOU USE A DIFFERENT WORD

Page 4: [6] Deleted **govi** **10/8/2004 6:47:00 PM**

processor please respect the recommended format of the templates.

CHECKLIST FOR ELECTRONIC PUBLICATION

Use only Times (bold or italic) and Symbol fonts for text—10pt minimum.
Figures should be in Times (bold or italic) and Symbol fonts when possible—6pt minimum.
Check that the PDF prints correctly.
Check that there are no page numbers.
Check that the margins are correct on the printed version. There may be differences of ±1 mm on the margins from one printer to another.

ACKNOWLEDGEMENTS

This template has been adapted from one of those published on the JACoW web site. We would like to thank the authors of those templates: J.Poole, C.Petit-Jean-Genaz, and P. Lucas.

Page 4: [7] Deleted **Alberto Aimar** **8/6/2004 9:45:00 AM**

the PDF file of the paper contribution.

Page 4: [8] Formatted **govi** **4/21/2006 5:18:00 PM**

Heading 2,Section Heading, Adjust space between Latin and Asian text, Adjust space between Asian text and numbers

Page 4: [9] Deleted **govi** **10/8/2004 11:35:00 AM**

LaTeX, Microsoft Word and OpenOffice. Authors are advised to use the templates provided. They are designed to help authors format their papers correctly.

General Layout

These instructions are a typical implementation of the requirements. Manuscripts should be prepared for one side of the paper and have:

A4 sheet paper size (21.0 cm × 29.7 cm; 8.27 in × 11.69 in)
Single-spaced text in two columns of 82.5 mm (3¼ in) with 5.3 mm (0.2 in) separation.

THE TEXT LOCATED WITHIN THE MARGINS SPECIFIED IN TABLE 1 TO FACILITATE ELECTRONIC PROCESSING OF THE

Page 4: [10] Deleted **govi** **10/8/2004 6:47:00 PM**

PDF file.

Table 1: Margin Specifications

Margin	**A4 Paper**
Top	37 mm

Bottom	19 mm
Left	20 mm
Right	20 mm

The layout of the text on the page is illustrated in Figure 1. Note that the paper's title and the author list should be the width of the full page. Tables and figures may span the whole 170 mm page width, if desired (see Fig. 2).

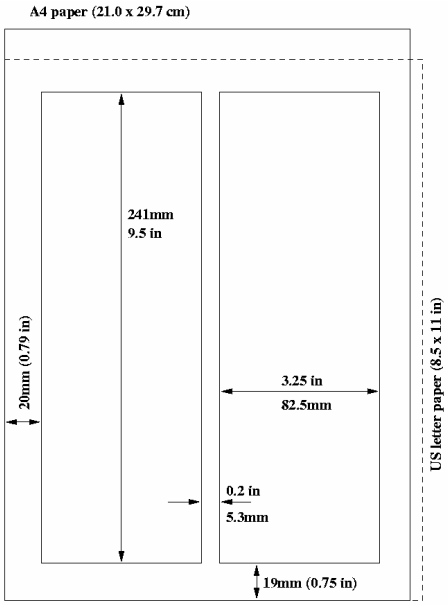
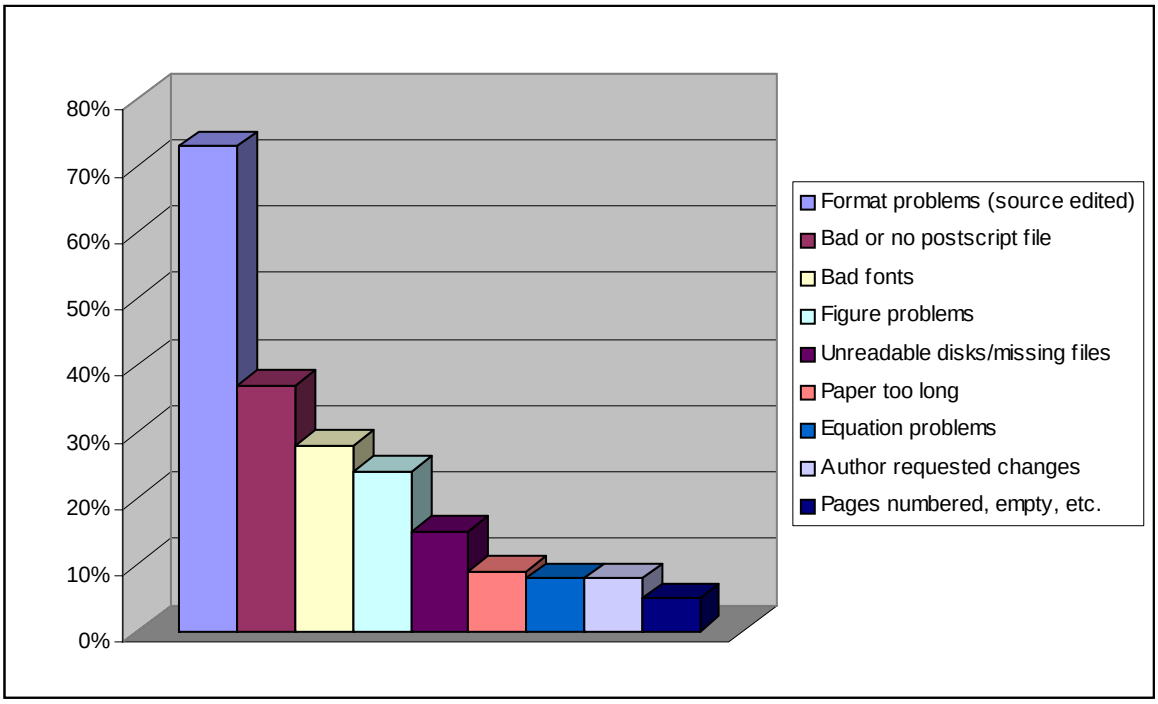


Figure 1: Layout of papers.

Fonts

In order to produce good Adobe Acrobat PDF files, the editorial staff asks authors to use only Times (in bold or italic) and Symbol from the standard postscript set of fonts.



Title and Author List

The title should use 14pt bold uppercase letters and be centred on the page. The names of authors and their organisations/affiliations and mailing addresses should be grouped by affiliation and listed in 12pt upper and lowercase letters. The name of the submitting or primary author should be first, followed by the co-authors in alphabetical order by affiliation.

Section Headings

Section headings should not be numbered. They should use 12pt bold uppercase letters and be centred in the column. All section headings should appear directly above the text—there should never be a column break between a heading and the following paragraph.

Column Break

Subsection Headings

Subsection headings should not be numbered. They should use 12pt italic letters and be left aligned in the column. Subsection headings should appear directly above the text—there should never be a column break between a subheading and the following paragraph.

Paragraph Text

Paragraphs should use 10pt font and be justified (touch each side) in the column. The beginning of each paragraph should be indented approximately 3mm (.13 in). The last line of a paragraph should not be printed by itself at the beginning of a column nor should the first line of a paragraph be printed by itself at the end of a column.

Figures, Tables and Equations

Place figures and tables as close to their place of mention as possible. Lettering in figures and tables should be large enough to reproduce clearly. Use of non-approved fonts in figures can lead to problems when the files are processed. Please use the approved fonts when possible [2].

All figures and tables must be given sequential numbers (1, 2, 3, etc.) and have captions (10pt font) placed below figures and above tables being described. Captions that are one line should be centred in the column, while captions that span more than one line should be justified.

Figure 2: Example of a full width figure showing the distribution of problems commonly encountered during paper processing.

TEXT SHOULD NOT BE OBSCURED BY FIGURES. FOR MORE INFORMATION ON WORKING WITH FIGURES IN MICROSOFT WORD OR OPENOFFICE (INCLUDING HOW TO INSERT THEM IN THE MOST EFFICIENT MANNER), SEE THE RESPECTIVE

Page 4: [11] Formatted **govi** **10/15/2004 10:19:00 AM**

Normal, Don't adjust space between Latin and Asian text, Don't adjust space between Asian text and numbers

Page 4: [12] Formatted **govi** **10/15/2004 10:19:00 AM**

Normal, Don't adjust space between Latin and Asian text, Don't adjust space between Asian text and numbers

Page 4: [13] Formatted **govi** **10/11/2004 10:14:00 AM**

Font: Times New Roman, English (U.S.)

Page 4: [14] Deleted **govi** **10/11/2004 10:14:00 AM**

A. Name and D. Person, Modern Editor’s Journal 25 (1997) 56.

Page 4: [15] Deleted **govi** **10/11/2004 10:25:00 AM**

A.N. Other, “A Very Interesting Paper,” EPAC’96, Sitges, June 1996, p. 7984.

Page 4: [16] Formatted **govi** **4/21/2006 5:33:00 PM**

Font: 10 pt, Font color: Auto