A DATA TREATMENT TOOL FOR A SATELLITE THRUSTER TEST FACILITY

Geraldo L. da S. Ribeiro
geraldo-ribeiro@uol.com.br
D. Bastos-Netto
demetrio@lcp.inpe.br
National Institute for Space Research (INPE), Cachoeira Paulista, São Paulo, Brazil

This work introduces and discusses CATPA (Computer Aided Thruster Performance Analysis), a user friendly data treatment software specially developed for INPE's Satellite Thruster Test Facility. This facility is a ground-based structure that can emulate the space environment for testing and qualifying satellite thrusters and satellite propulsive loops. A structure like this requires the prompt availability of data treatment tools which must be extensible to encapsulate the particular needs of the customers. The original data treatment package supplied by the test bench development team needed a strong work to customize the available features. However several constraints were imposed for us, like: a) support for low amount of data; b) large time slices were required for processing the acquired data; c) poor final output quality; d) specific hardware requirements; e) non portability; f) specific language used for coding the scripts; etc. The CATPA came up to override these constraints and to make the data treatment easier and much more powerful. It uses modern concepts like design patterns and standard template library, making the reuse of code and the rapid application development easier to be reached.

Introduction

This work started out with the intention of using the supplied scripts to treat the acquired measurements. However, after some testing it was concluded that these scripts were not fast enough non flexible enough for existing needs. This led to the coding of a new treatment tool called CATPA.

Development architecture

The development architecture is an AMD K6-II 300 MHz machine with Debian GNU/Linux 3.0 operating system running a MySQL database engine version 4.1.

The current release of the CATPA suite only supports the MySQL database engine. But it can be easily extended to a large amount of database engines.

The Scopix

The Scopix software is a data acquisition tool with real time visualization capabilities. The applicability of this software tool at test benches was due to its convenient features. Its interface is easy and powerful. The user interface leads the operator in a real way: a) the sensors creation; b) the cables configurations; c) the test definition; etc. However the data stored into Scopix's disk must be transfered to another machine for external treatment.

Data Transfer Technique

The first problem to be solved was: How to transfer the acquired data to another machine?

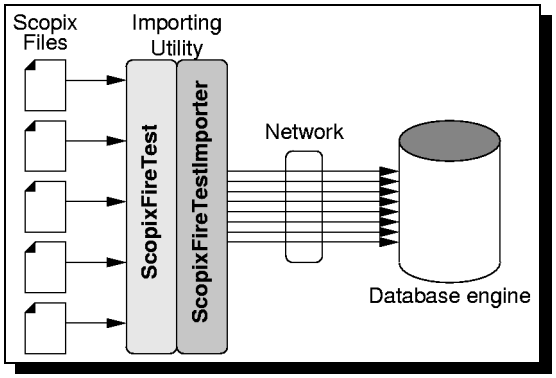

Figure 1 The importing scheme.

Step number zero was to define "*another machine*". It was chosen the other machine to be a host which runs under an unknown operating system and a well specified database engine.

Step number one was to define "*what to transfer*". To solve this step a complete and well substantiated know-how about the Scopix files format were needed. Once the operator possesses the informations above he is able to code a transfer driver.

The importing driver is divided in two parts (as shown in figure 1): a Scopix reliant part and a database engine reliant part. To supply the first part it was coded a *C++ class* called `ScopixFireTest`. This *C++ class* handles all features of a Scopix fire test. The `ScopixFireTest` *C++ class* works as an interface element between the Scopix files and the importer routine. The second part of the transfer driver inherits

```
$ catpaScopixFireTestImporter

Usage: catpaScopixFireTestImporter [options]

where [options] are:

  -c --campaign     Campaign name
  -t --test         Test name
  -u --user         Username
  -p --password     Password
  -v --verbose      Verbose mode
  -h --help         Print this help
```

Figure 2 The `catpaScopixFireTestImporter` command usage.

`ScopixFireTest` capabilities adding the import function.

The import function is coded using a specific database engine set of commands. This approach is needed because the import function demands a strong number of database engine calls. The change of the database engine implicates in the re-coding of the import function. This re-coding is needed to keep up the *high performance* of the application. This approach was chosen due to the strong loop needed to import all the measurements records. But all other database engine calls are made using a generic set of SQL commands thus keeping the code portability.

The importing driver was implemented with command line behavior to be used in batch processing mode. Its usage is shown at figure 2.

Interface

The graphical user interface is quite intuitive, resembling a pair keyboard/screen terminal (see figure 3). The user pushes the buttons on the lower part of the application and the results are displayed in the log area. These log messages are stored on the database, and copied both to the standard output device and to a report file.

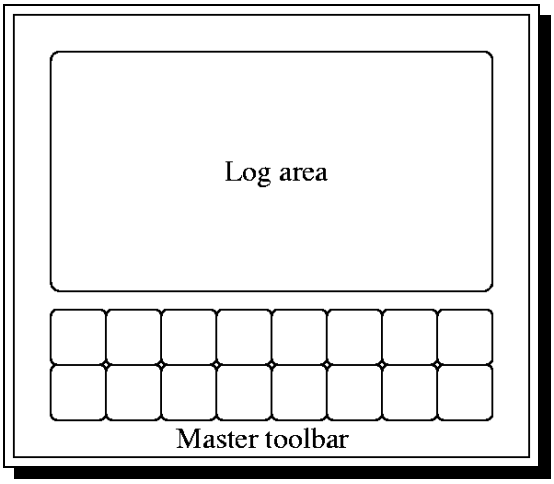

Figure 3 The CATPA main window scheme.

The main window, as shown at figure 3, consists basically in a log area (`LogWindow` *C++ class*) and a toolbar (`MasterToolbar` *C++ class*). The `LogWindow` *C++ class* handles the output messages by scrolling its history.

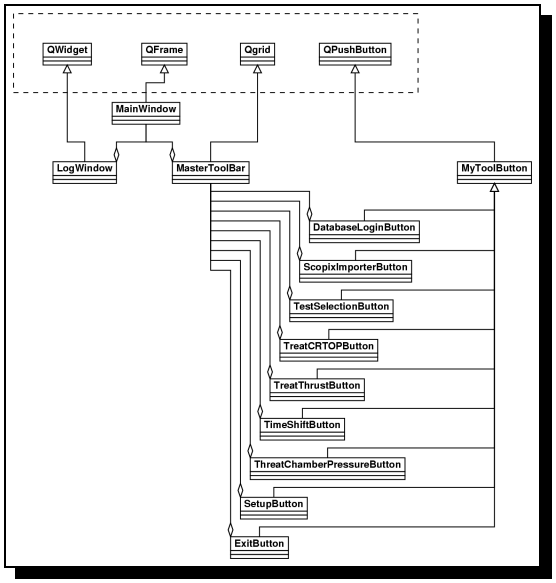

Figure 4 Partial *C++ class* diagram of CATPA.

All new treatment routines must be placed into the `MasterToolBar` by inheriting `MyToolButton` *C++ class*. Figure 4 shows the widgets relationship.

Default data treatment

The standard data treatment is composed by:

- Time adjustment;
- Pulse description;
- Mean thrust calculation;
- Mean combustion chamber pressure calculation;
- Ignition time delay calculation;

Time adjustment

At the beginning of the fire test two delays occur: a) The acquisition system pre-trigger; and b) The control-Command pre-trigger. These delays are necessary only to supply the start conditions. But its time slice must be suppressed for the correct data analysis to take place.

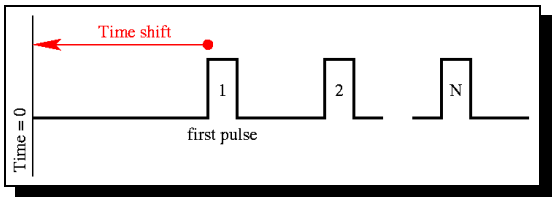

Figure 5 Time adjustment scheme.

By using the PLC (Programmable Logic Controller) command signal the CATPA searches for the starting of the first pulse and then it decrements the time scale

of all measurements. This decrement puts the real test start at time zero. A scheme for the time adjustment procedure is shown at figure 5.

Pulse boundaries

The initial treatment step is to locate the pulses boundaries. Figure 6 displays a schematic pulse boundaries description. These limits are nedeed to select correctly selection of the sampling region.

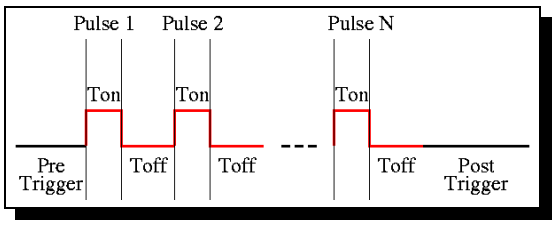

Figure 6 Pulse boundaries.

Analysis span

For short pulses, some assumptions are made: The solenoid valve stats their opening process upon receiving the ON signal from the PLC within a very short time delay (circa 1 ms). The solenoid valve is assumed to be fully opened with *70%* of the total allowed current. In this case the best region for the data analysis is the second half of the pulse, as shown in figure 7. By choosing the second half of the pulse, a quasi steady state can be assumed for calculation purposes. In the case of long pulses the analysis span is limited by a variable called `Treat/AnalysisSpan` (*3s* by default).

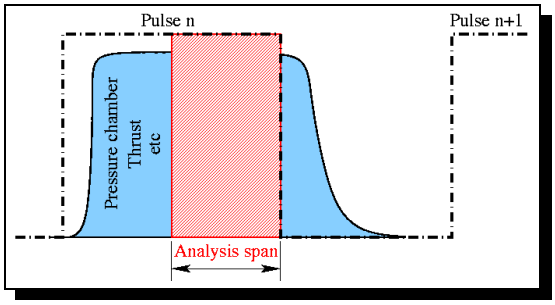

Figure 7 Analysis span.

Mean values

The mean values of thrust, pressure chamber, propellant flow rate, ..., used for calculation of several performance parameters, are estimated by using data sampled from the analysis span, as shown in figure 7.

Ignition time delay

Figure 8 shows the PLC command, the propellant solenoid valve current and the combustion chamber pressure behaviors. Notice the delay between the starting of the PLC pulse and the starting of the pressure pulse.

The ignition time delay is considered to be the time interval between the PLC starting command and a

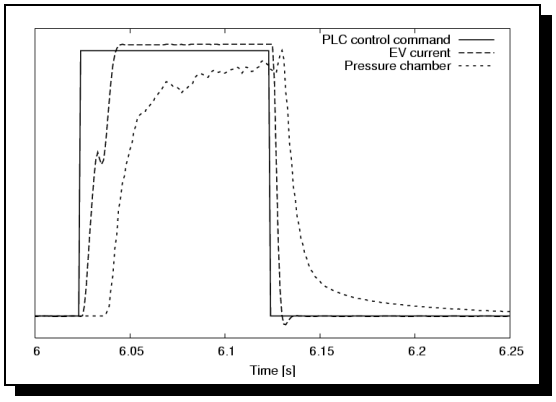

Figure 8 Analysis span.

sensible change in the combustion chamber pressure. For estimation of the ignition time a tendency curve is required. This tendency curve is calculated using the sampled data between t_0 and t_1 (figure 9). t_1 is the starting time of the PLC pulse. The t_0 value is given by $t_0 = t_1 - t_{ts}$, where t_{ts} is the value of the `Treat/TendencySpan` variable.

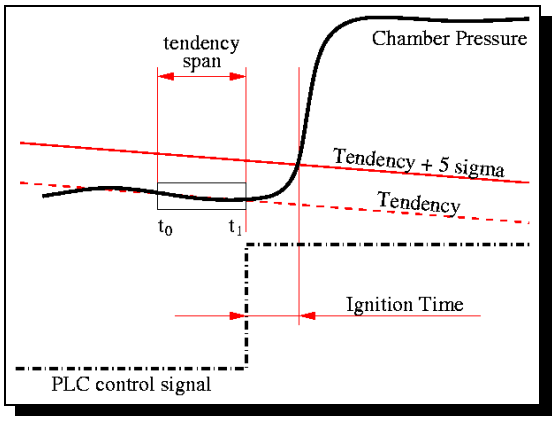

Figure 9 Ignition time calculation scheme.

Database scheme

The CATPA uses a multi-platform SQL database engine allowing the enhancement of the actual application features and the coding of a totally new application quickly, because no specific technology is required. The same application can run on several database engines with a single mouse click. The default database engine is the MySQL. This database engine was chosen due to its high performance characteristics. The database scheme for the CATPA application is shown in figures 10 to 13.

The logMessages table

The `logMessages` table (figure 10) holds all messages generated by the CATPA, enabling the user to inspect the action history.

The tests table

The `tests` table (figure 11)holds the main description of the fire test.

```
CREATE TABLE logMessages
(
    id                          INTEGER AUTO_INCREMENT
                                NOT NULL PRIMARY KEY,
    login                       VARCHAR(50),
    date                        DATE,
    time                        TIME,
    level                       INTEGER,
    message                     VARCHAR(255)
);
```

Figure 10 The `logMessages` sql table scheme.

```
CREATE TABLE tests
(
    id                          INTEGER AUTO_INCREMENT
                                NOT NULL PRIMARY KEY,
    campaign                    VARCHAR(50) NOT NULL,
    test                        VARCHAR(50) NOT NULL,
    date                        DATE,
    time                        TIME,
    minSamplingTimeStep         REAL,
    maxSamplingRate             INTEGER,
    recordedTime                REAL,
    bench                       VARCHAR(50),
    propellant                  VARCHAR(50),
    throatDiameter              REAL,
    exitDiameter                REAL DEFAULT NULL
);
```

Figure 11 The `tests` sql table scheme.

The `measurements` table

The measurements table (figure 12) hold informations about the acquired data of all fire tests.

```
CREATE TABLE measurements
(
    id                          INTEGER AUTO_INCREMENT
                                NOT NULL PRIMARY KEY,
    campaign                    VARCHAR(50) NOT NULL,
    test                        VARCHAR(50) NOT NULL,
    name                        VARCHAR(50) NOT NULL,
    unit                        VARCHAR(50),
    samplingRate                INTEGER NOT NULL,
    timeStep                    REAL NOT NULL,
    label                       VARCHAR(255),
    tableName                   VARCHAR(50) NOT NULL,
    type                        CHAR(5)
);
```

Figure 12 The `measurements` sql table scheme.

The `pulses` table

The `pulses` table (figure 13) holds information about each pulse of all fire tests.

```
CREATE TABLE pulses
(
    id                          INTEGER AUTO_INCREMENT
                                NOT NULL PRIMARY KEY,
    campaign                    VARCHAR(50) NOT NULL,
    test                        VARCHAR(50) NOT NULL,
    number                      INTEGER NOT NULL,
    start                       REAL NOT NULL,
    end                         REAL NOT NULL,
    onTime                      REAL NOT NULL,
    offTime                     REAL NOT NULL,
    startAnalysis               REAL NOT NULL,
    endAnalysis                 REAL NOT NULL,
    thrust                      REAL NOT NULL,
    chamberPressure             REAL NOT NULL,
    ignitionTime                REAL NOT NULL
);
```

Figure 13 The `pulses` sql table scheme.

Hard disk usage

The CATPA has no data size constraint, extending the old one Giga-byte limitation to several Tera-bytes of data files. The old application uses the same file for handling all fire tests. The opposite approach is employed by the CATPA enabling the developer to use up to date technologies to improve the speed analysis. The use of one separated table (file) for each acquisition rate group provides a lot of benefits for tasks such as incremental backups, data exchange and data conversion which are made much simpler.

The output "factory"

This application supplies templates for coding and adding new features. The report output format is controlled by a factory design pattern allowing the developer to code drivers for any other formats.

The operation system

CATPA is developed under Linux, but it is set-up to be highly portable. As a result, it runs on most other Unix flavors as well. For graphical user interface it was used the multi-platform toolkit called Qt. The programming language is C++.

For a better use of the available hardware on the test facility the CATPA software was written to be modular. This feature enables it to get raw data from the Scopix workstation at Host A and to put it into a database server located at Host B. The CATPA main application (located at Host C) opens a data stream between hosts B and C, performs the calculation and puts the results in another machine (Host D). The opposite approach is also possible, i.e., the concentration of all operations in a single machine.