To understand the listing of the tool `asverif/asrest`

Summary:

The tool (`asverif/asrest`) of checking of the sources (before their integration in the sources official) produced a listing sometimes long and complex to understand. The goal of this document is to describe the general structure of the listing as certain details which can divert the user of this tool.

Contents

1 General structure of the listing

The tool (which we will call more simply asverif thereafter) is actually a succession of elementary tools which are connected the ones with the others. Each elementary tool has as a function to control part of the sources (FORTRAN, catalogues, tests,…). The listing produced by asverif is thus the concatenation of the listings produced by these elementary tools.

The sequence of the elementary tools is not completely arbitrary. For example, the edition of the links cannot be done before compilation.

Sometimes, certain errors detected by the tool prevent it from continuing the analysis; for example, if modified FORTRAN (or addition or removed) is unacceptable (by the tool), there will be no compilation of the catalogues because this compilation must be done with the achievable aster which depends on modified FORTRAN. The listing will then be truncated.

In the following example, we tried to connect the most possible elementary tools. For that, we used asverif on a “overload” of the code including:

- routines FORTRAN
- catalogues of order
- catalogues of elements
- sources python
- suppressions of source (unigest)

The listing will have the following general form then:

```
=== Checking of the source code of J.PELLET (vabhhts)
================================================================================
…   <VERIF_1> (checking of the file "histor")

*** Checking of source FORTRAN
************************************************************
…   <VERIF_2>

*** Checking of source CATALOGU
************************************************************
…   <VERIF_3>

*** Checking of source CATALOPY
************************************************************
…   <VERIF_4>

*** Checking of the PYTHON source
************************************************************
…   <VERIF_5>

*** Checking of source UNIGEST
************************************************************
…   <VERIF_6>

=== Restitution of the source code of J.PELLET (vabhhts)
================================================================
…   <VERIF_7>

=== Compilation of sources FORTRAN in majobj                 ===
================================================================
…   <VERIF_8>  (compilation f90 + CRS tool)
```

```
    Compilation of sources FORTRAN with /aster/outils/g77.csh
================================================================
…   <VERIF_9>  (compilation g77)

=== Checking of sources FORTRAN with CRP                ===
================================================================
…   <VERIF_10>

=== Edition of the links of the sources FORTRAN C and CAL          ===
================================================================
…   <VERIF_11>

=== Construction of the catalogue of orders python
================================================================
…   <VERIF_12>

=== Compilation of the catalogue of elements
================================================================
…   <VERIF_13>
--- End of asverif.exesh with the code of return: 4
----------------------------------------------------------
--- CODE RETURN = 4
----------------------------------------------------------
----------------------------------------------------------
--- DIAGNOSIS JOB: ERREUR_AGLA
----------------------------------------------------------
```

Each part of the listing (here called "<VERIF_n> ") is produced by an elementary tool. It is introduced by a character string which is clean for him. For example, the checking of the file "histor" (<VERIF_1>) is introduced by the chain "=== Checking of the source code of J.PELLET (vabhhts)"

In the following paragraphs, we will reconsider certain parts of this listing, those concerning the sources FORTRAN.

2 What is necessary to read in this listing?

All in all, the listing comprises many useless impressions (those which say that “all is well”). This information does not facilitate the task of reading, but they make it possible to check that the sources taken into account are well the goods.

Therefore, all the listing should not be read! Information not to miss is the fatal errors and alarms.

The fatal errors (those which will make impossible the restitution) begin all with the chain "(F/U-". For example:

```
(F/U-015-(a)): The /cluster/members/member2/tmp/519552/vabhhts/histor file
is not in conformity with the groundwork
(F/U-h01b): the documentary keys X0.00.00 are prohibited.
(F/U-046): acou_face3: IMPOSSIBLE SUBCATALOGUE SUPPRESSION
(F/U-091): ERROR (S) and WARNING (S) FORTRAN (detected by g77):
```

Alarms begin all with the chain "(A/U-". For example:

```
(A/U-040): MODIFICATION OK of: calculel/calcul.f
```

```
(A/U-558): grandeur_simple: modification of a unit of which the person in
charge is: JMBHH01
(A/U-043): debcal: FORSUPPR ROUTINE OK
(A/U-044): sslx101a: SUPPRESSION CAS-TEST OK
```

These examples show that unfortunately, these “alarms” are not always alarms!

To help the user to find itself there, the tool `asverif` fact a summary of alarms having involved a code return not no one. Caution: this summary is produced only if no fatal error is detected.

Example:

```
################################################################################
### Summary of Alarms involving a code return = 2:
(A/U-h10):  Modification of the document: U4.02.01

(A/U-222): calculation: Modification of the map SHEET
           The errors preceded by has were additions in the map
           The errors preceded by an S were removed map
WITH CRP_20 CALCULATION
################################################################################
```

2.1 Summary

When one uses one of the tools of `the agla (asverif/asrest/pre_eda)`, the chain should initially be searched `"(F/U"` in the listing. This chain “points” the fatal errors. When all the fatal errors are corrected, it is then necessary to look at the summary of Alarms.

Note:

For some time, a new checking is made concerning coherence between the identifieurs of message used in the sources (for example: `ALGELINE_12`) and the presence of these messages in the catalogues of messages `(bibpyt/Messages/*.py)` . This checking (at the end of the listing) message does not emit unfortunately `"(F/U"` in the listing.

Detected problems: “missing” message or “unutilised” message appears in the following form in the listing:

```
Unutilised messages:
CALCULEL4_4 PREPOST6_37
Non-existent messages:
PREPOST6_36
```

3 Problems involved in FORTRAN

Sources FORTRAN are those which pose the most problems with the users of agla. The reason is the multiplicity of the tools which theirs are applied:

- script of control of FORTRAN
- compiler (S)
- tool for checking of the standard ANSI77 (CRS)
- tool of intercompilation (CRP)
- linkage editing

We will illustrate the listing of these tools.

3.1 Script of control of FORTRAN

This script is carried out at the beginning of the tool. Its listing is in the part <VERIF_2>. Several checks are carried out:

- that the source resembles FORTRAN
- that the source does not contain illegal characters (or disadvised) or too long lines
- that the source comes well from the official last version (checking of the map "C MODIFICATION")
- that the user did not modify the cards "C SHEET" (one will understand this sentence after having read the paragraph concerning the tool CRS)

```
--- Checking which the source is of type FORTRAN

/tmp/519797/aster/eda/vabhhts/identifi/fortran contains source FORTRAN

--- Checking which the source does not contain
--- awkward characters
  17: C WITHOUT ANY WARRANTY; WITHOUT EVEN THE IMPLIED WARRANTY OF  MERCHANTAB
  17/+: ILITY GOLD FITNESS FOR WITH PARTICULAR PURPOSE. SEE THE GNU  GENERAL
PUBLIC LICENSE FOR MORE DETAILS.


(A/U-014): Preceding lines (of the /tmp/519797/aster/eda/vabhhts/identifi/fortran
file)
        : are preceded by their number in the file and a diagnosis:
            num_ligne/-: the first 72 characters of a line (FATAL ERROR)
            num_ligne/+: the continuation of the preceding line        (FATAL
ERROR)
            num_ligne/!: contains disadvised characters   (ALARM)
            num_ligne/?: contains ILLEGAL characters      (FATAL ERROR)


(F/U-014): The /tmp/519797/aster/eda/vabhhts/identifi/fortran file contains:
         :     ILLEGAL characters
         :     and/or of the TOO LONG lines

--- Checking of coherence with the version of reference (NEW7)

(A/U-222): calculation: Modification of the map SHEET
           The errors preceded by has were additions in the map
           The errors preceded by an S were removed map
WITH CRP_20 CALCULATION
```

3.2 Compiler (S)

Only the fatal errors of the compiler are to be taken into account. The possible ones "warnings" are not intercepted byagla. On the machine currently used byagla (in August 2003: alphaserver), one compiles 2 times the sources using 2 different compilers f90 (<VERIF_8>) and g77 (<VERIF_9>).

extract of <VERIF_8> :

```
================================================================
=== Compilation of sources FORTRAN in majobj                 ===
================================================================

(F/U-090): ERROR (S) and WARNING (S) FORTRAN:
+ f90 - v - C - fast - i8 - r8 - omp
/tmp/519797/tmp_asverif/bibfor/calculel/…
f90: Warning: /tmp/519797/tmp_asverif/bibfor/calculel/calcul.f, line 268:
     Variable IACHIX is used before its been worth has been defined
            EXICH = ((IPARIN.GT.0) .AND. ZL (IACHIX-1+IPARIN))
-----------------------------------------^
```

Note:

asverif be based on the code return of the ordering of launching of the tool for compilation to transmit the error message ((F/U-090) : ERROR (S) and WARNING (S) FORTRAN). It is thus necessary then to consult the messages transmitted by the tool for compilation.

3.3 Tool for checking of the standard ANSI77 (CRS)

The tool CRS in the public domain (it was recovered actually acts of a hardly modified version of the tool ftncheck). The achievable one of this tool produces a listing (in <VERIF_8>) being able to contain a large number of types of errors or of warnings. Each type of message is preceded by a label of the form CRS_nnn ; for example:

```
CRS_206 routine CALCULATION
"/tmp/519808/tmp_asverif/bibfor/calculel/calcul.f",
        line 93 collar 15: Warning: not standard: (declaration except
standard: REAL*8,…)

CRS_513 routine CALCULATION: Warning: Variables used before set: IACHIX
```

The "trap" of this tool is that in the printed messages (in English), it uses the words "error" and "warning" whereas for the tools ofaglhas these words are not inevitably synonymous with fatal error or alarm.

There is "Warnings" CRS who involve fatal errors ofagla and of "errors" CRS of no importance!!!

The principle of the use of CRS inagla is the following:

- The program is carried out CRS and his listing is recovered: l1
- One looks at so in the listing l1, there exist lines starting with labels "CRS_nnn" searched byagla (note: certain messages CRS_nnn are completely ignored; for example those concerning the types except standard REAL*8 or COMPLEX*16).
- In the lines selected, one withdraws the errors CRS_nnn who are tolerated (mechanism of "C CRS_nnn SHEET" described below).

- If there remain lines, the following text is printed:
- ==⇒ Compilation refused for Code_Aster because of the errors FORTRAN CRS following (without map 'C SHEET') :
- CRS_513 CALCULATION
- Then the rough listing is printed l1

3.3.1 Mechanism of "C CRS_nnn SHEET"

agla consider that certain "errors" of CRS can suffer from exceptions. For example, the prohibition of equivalence between numeric variables and type CHARACTER impossible the use makes of JEVEUX. So that an exception is allowed by the tool, the user must indicate in his source which he violates deliberately the rule of programming. For that, he will write for example in one of his routines: "C SHEET CRS_513"

3.3.2 Summary

To exploit the result of the tool CRS, the chain should be searched "==⇒ refused Compilation…" If it is present, only the messages should be looked at CRS_nnn who are indicated to the lower part.

3.4 Tool for inter-compilation (CRP)

The tool CRP function in the same way as the tool CRS. It also accepts exceptions managed by the cards "C CRP_nnn SHEET".

The specificity of this tool is to check the inter-compilation of the sources FORTRAN. I.e. for example:

- to check that the number and the type of the arguments at the time of a call to a procedure are same as those definite at the time of the definition of the procedure,
- to check the coherence of the various occurrences from one COMMON,
- …

```
Example of listing of CRP  (<VERIF_10>):
================================================================
=== Checking of sources FORTRAN with CRP                 ===
================================================================
(F/U-222): Error (S) detected (S) by crp:
===> Errors crp and inter-compilation:
CRP_150 CALCULATION
CRP_3 CALCULATION
```