CLRC

CLRC

Software Quality Assurance for Fortran 90 - A Survey of Available Tools

R F Fowler and C Greenough

Software Quality Assurance for Fortran 90 - A Survey of Available Tools

R.F. Fowler and C. Greenough

Computational Science and Engineering Department
CLRC Rutherford Appleton Laboratory,
Chilton, Didcot OX11 0QX, UK

Email: r.f.fowler@rl.ac.uk or c.greenough@rl.ac.uk

This report is available from http://www.cse.clrc.ac.uk/Activity/HPCI

July 2, 1999

Abstract

This report provides a survey of some of the software tools currently available to assist in the development of Fortran programs using the new 1990 and 1995 standards. A few tools also address the development of High Performance Fortran (HPF). This report is particularly concerned with tools which will enhance the quality of software developed. We have not looked at tools which focus on the high level design and requirements analysis, since these are often common to many languages. Instead we have concentrated on tools that help in the detection of programming errors and highlight poor practice, such as "spaghetti" style coding. At present, there appear to be few such tools available for the new Fortran standard. However, many Fortran 90 compilers can provide quite extensive compile and run time checks and some examples of these are also discussed.

Keywords: software quality, Fortran 90, Fortran 95, High Performance Fortran

Contents

1 Introduction

The Fortran 90 standard has been available for many years but its uptake has been rather slow, perhaps due to the time taken to develop high quality optimising compilers and competition of other languages such as C++. This situation is changing now as good compilers are becoming more common and there is a lot of interest in exploiting the parallelism of multiprocessor machines through High Performance Fortran (HPF) and OpenMP directives.

Fortran 90 offers many significant advantages over the previous standard. Among these dynamic memory management, abstract data types, data hiding, array notation and explicit subprogram interfaces are some of the most important. When used correctly these new features should allow the development of better quality software, as operations can be expressed more clearly and the compiler can perform many more checks.

Nevertheless it is just as easy to produce poor quality software in Fortran 90 as in any other language and some of the added features, such as pointers, have potential to cause new types of errors. Hence there is a need for software tools to aid the development of scientific and engineering applications in Fortran 90.

This document contains a short description of some of the more useful software tools that are currently available to assess and improve the quality of Fortran 90 software. Some tools are also applicable to Fortran 95, the latest revision of the standard, and to HPF. We have not considered tools which focus exclusively on the high level design and requirements analysis. These are of course important in the production of large software packages but, since they are often common to many languages, it is not appropriate to include them in a review of Fortran tools.

Software quality tools are part of the broader class of software re-engineering tools. A fairly comprehensive review of such tools, covering many programming languages, has been placed on the Web [5]. However, the review gives very few details on specific Fortran 90 products as it was compiled in 1995. Some extracts from this report, mainly about Fortran 77 tools, are listed in the appendix to this document.

The software tools we shall consider may be roughly divided into three classes, though there is a degree of overlap between them. These are:

- Analysis tools, which check source code for adherence to standards, consistency of argument lists and similar problems. They may also provide measures of software complexity;

- Tools which provide some form of reverse engineering and re-documentation capabilities to extract call graphs and other high level design data from the source code;

- Conversion tools, which restructure the software. This includes converting old software to use the more readable and safer syntax of Fortran 90;

- Fortran 90 compilers and Integrated Development Environments (IDEs). Many compilers now include sophisticated checks built into them and may offer things such as inter-procedure analysis which was previously only available in separate tools. Debuggers may also provide checks on memory leaks and variables used before set, which is only fully possible at run time. Some vendors provide IDEs which closely couple software analysis and browsing tools with the compiler.

Since there is such a large number of these tools, this review cannot include them all or even provide comprehensive details of those that we do discuss. However, it should give an idea of the range and capabilities of tools available at the time of writing.

2   Software Quality

Software quality is not a term that has a precise, "scientific" definition. Nevertheless it is vital that software developed for research into scientific and engineering problems should strive to be of high quality for several reasons:

- If the quality of the software used to produce a new computational result is questionable, then the result must also be treated with caution. Just as results from a poorly designed experiment are of little use, computational results need to be based on sound theory coupled with quality software;

- A research code may be developed by several different students or staff at different times. Poor application of software quality control techniques can lead to programming errors, wasted development time and dubious results;

- High quality software can be made available to other research groups, or included in commercial products.

Production of high quality software requires systematic design and development coupled with comprehensive testing. Clearly this can be at odds with scientific investigations which often require rapid results to prove the usefulness, or otherwise, of a new method. However it is vital to strike the right balance between the need for rapid development of a prototype code and the requirement of trustworthy results and re-usable code. Improving the quality of poorly written code can be a very time consuming process, and it may prove easier to start again from scratch. It is much more efficient if software quality control is applied from the beginning, rather than trying to retro fit it to existing code.

The need for quality assurance is underlined by research such as that due to Les Hatton [1]. Hatton has analysed a large number of commercial scientific software packages written in C and Fortran 77. The Fortran 77 codes had an average of 12 statically detectable faults per 1000 executable statements (these were found using the QA Fortran static analysis program). In run time tests of nine software packages for seismic data analysis, which nominally implement exactly the same mathematical algorithms, substantial variations in the results were found. These differences are sufficient to cast doubt on some new analysis methods which require high accuracy in the processed data.

Another investigation of software quality in Fortran codes has been made by Hopkins [6]. He has investigated how the quality metrics of some well known public domain packages and published algorithms have changed with time. The most worrying aspect of this work is that there seems to be little improvement in the quality measures of many published Fortran algorithms over the past twenty years.

Software tools can be useful is the assessment of a software product in several ways:

- Detection of software defects that are not usually reported by compilers, such as argument mismatch between program units and use of unassigned variables;

- Objective measures of programming style and complexity, e.g. a program unit has an excessive number of source code lines or has so many independent paths through it to making testing difficult;

- Instrumentation of the code to identify parts that have not been exercised by a test set;

- Restructuring to improve readability and maintainability;

- Extraction of software design and data flow from the source code to aid understanding and re-documentation.

To be really effective they should be used frequently through the development process so that errors and deviations from good programming practice can be identified early on. Application to existing software can be helpful in identifying problems and hidden faults. However, they cannot be expected to transform poorly-designed software into good software. Careful design and testing are still vital in the development process.

A useful guide to coding and documenting Fortran 90 software has been placed on the Web by the UK Meteorological Office, at:

    `http://www.meto.gov.uk/sec5/NWP/NWP_F90Standards.html`

While not all the rules and suggestions in this document will be applicable to every software developer, it could be useful as the basis of a local coding standard.

Another useful website for Fortran related software is the Fortran Market Place:

    `http://www.fortran.com/fortran`

This has many links to information on Fortran 90 compilers and some public domain software such as Ftnchek (error checking for Fortran 77) and convert.f90 (converts fixed format to Fortran 90 free format).

3 Software Tool Capabilities

In this section we outline some of the capabilities that are available in software tools that can be used for quality assurance purposes.

3.1 Static Error Detection

Static errors may be detected by programs which provide *lint*-like checking for Fortran 77. Possible tests include:

- Argument checking - are the number and type of arguments consistent between the declaration of a subprogram and each call;

- Common block checking - warnings can be generated for named common blocks of different lengths, and related potential problems;

- Use of implicit typing - though standard conforming, implicit typing often leads to errors;

- Unused items - variables and sections of code that are never used are reported as these are at least a source of unnecessary clutter and may indicate programming errors;

- Implicit type truncations, such as copying of real to integer, may lead to unwanted truncation.

In Fortran 90 all the above programming errors can be made since Fortran 77 is a subset of the language. However careful use of new features, such as explicit interfaces to all subprogram interfaces and replacing common blocks with modules, should enable the compiler to detect some of the above problems. On the down side Fortran 90 does allow new errors to occur such as those associated with memory management. Static analysis should be able to detect some of these problems, for example allocatable and pointer variables that are used before allocation. Other errors, like references through a pointer to data that has been deallocated (a "dangling" pointer), may only be caught by detailed run time checks.

A system such as ftnlint, provided with SGI and Cray Fortran 90 compilers, is able to generate warnings for many of the common static errors listed for Fortran 77 style codes. It can also make global checks on module variables and parameters, argument INTENT statements, subroutine calls where explicit interfaces are required, and related problems.

3.2 Dynamic Error Detection

Many software errors can only be detected at run time, as was mentioned in the previous section. Some software tools have been developed which will instrument the source code to provide extra run time checks. Array bound errors and variables used before set could be detected in this way. However, these checks can often be more conveniently built into the compiler and/or its run time support system. The majority of Fortran 90 compilers now include array bound checking as an option. Some also have options to detect variables used before set and memory leaks. In some cases this additional support requires the use of an interactive debugging tool, which runs the executable program.

3.3 Code Restructuring and Transformation

A lot of old Fortran software is very poorly structured due to it having been developed originally as Fortran 66. A number of tools exist to transform and simplify such spaghetti code to Fortran 77 style code. Some tools are now becoming available to further transform Fortran 66 and 77 to take advantage of the new Fortran 90 syntax. Possible transformations include:

- Source code transformation from fixed format to the new free format;

- Reformatting and "pretty printing";

- Re-structuring to remove spaghetti code and use new constructs such SELECT, EXIT, CYCLE and array notation;

- Transformation of common blocks into modules, possibly with USE and ONLY statements to minimise potential name clashes;

- Generation of interface blocks for subroutines to allow better compile-time argument checking;

- Addition of appropriate INTENT statements to subroutine arguments.

An example of how one particular tool, Vast90, can translate code is shown below. The original Fortran 77 code is listed first, followed by the Fortran 90. Note that this example was produced using an old version of Vast90 with the default options, so it may not show the full capabilities of the current version, in which the transformation tool is now known as Vast77to90.

```
C f77
      SUBROUTINE MATMUL(A,A1,A2,L,N,M,NB)
C...............................................
      INTEGER NB,L,M,N
      DOUBLE PRECISION A(NB,NB),A1(NB,NB),A2(NB,NB)
C     Local
      INTEGER I,J,K
C                              Aik = A1ij . A2jk
      DO 10 I=1,L
        DO 20 K=1,N
          A(I,K)=0.0
          DO 30 J=1,M
            A(I,K)=A(I,K)+A1(I,J)*A2(J,K)
30        CONTINUE
20      CONTINUE
10    CONTINUE
      RETURN
      END

! f90
      SUBROUTINE MATMUL(A, A1, A2, L, N, M, NB)
!...Translated by Pacific-Sierra Research VAST-90 1.02X
      IMPLICIT NONE
!---------------------------------------------------
!   Dummy   Arguments
!---------------------------------------------------
      INTEGER L, N, M, NB
      DOUBLE PRECISION, DIMENSION(NB,NB) :: A, A1, A2
!---------------------------------------------------
!   Local   Variables
```

```
!-----------------------------------------------------
      INTEGER :: I, J, K
!-----------------------------------------------------
!                                    Aik = A1ij . A2jk
      DO I = 1, L
         DO K = 1, N
            A(I,K) = 0.0
            A(I,K) = A(I,K) + SUM(A1(I,:M)*A2(:M,K))
         END DO
      END DO
      RETURN
      END SUBROUTINE MATMUL
```

In this case the array notation has been used to remove the inner loop, and the outer two loops left in place. In this particular case the whole subroutine could better be replaced with the intrinsic routine MATMUL.

3.4 Software Structure Analysis

To aid understanding and maintenance of software it is useful to be able to extract call graphs and other diagrams of the structure. Variable cross referencing tables are also a useful tool for software maintenance.

A typical cross reference map produced by the Sun Fortran 90 compiler is shown below for one subroutine.

```
Cross-reference map:   13 Identifiers (74 References)
```

Type Kind		Usage	EqDSv	Offset	Home	Name	References							
Char		1-D Arr		1	Arg	ARGNAM	2	8	17-	29=				
Int	4	Scalar	–	0	Stack	IND1	14	20=	21+	22	23=	24	25	26
Int	4	Scalar	–	1	Stack	IND2	14	24=	25+	26=	29	30	32+	33
Int	4				Intrin	INDEX()	20	22	24	25				
Int	4	1-D Arr		2	Arg	LENARG	2	10	17=	30=	36=			
Int	4	Scalar		4	Arg	LENST	2	12	20	22	24	25		
Int	4				Intrin	LEN_TRIM()	17							
Int	4				Intrin	MIN()	36							
Int	4	Scalar	–	2	Stack	NEWIND	14	22=	23+					
Int	4	Scalar		5	Arg	NOARG	2	11	17	28=	29	30	36+	
Char		Scalar		3	Arg	STAMNT	2	9	20-	21	22-	24-	25-	27

Such listings can be used to understand where variables are modified and used and can also form part of the supporting documentation of the code. Similarly call graphs help to understand and document the control flow.

3.5 Software Metrics

Some software analysis tools go beyond checking the correctness and overall structure of a program and look at the complexity of the code within subprograms. Many different metrics have been defined which can be used to measure the quality of programs. These include things such as the static path count, the number of ways through a given subprogram (related to the Npath metric [2]) and the cyclomatic complexity metric due to McCabe [3]. Based on

programming experience, there are recommended maximum values for these measures, which indicate when a subprogram is poorly structured or is too convoluted to understand or test.

Using such metrics during the course of software development is important in highlighting excessively complex subprograms as they are being developed. The programmer can then restructure the code, perhaps splitting it into more modules. Application of a metrics analysis to existing software can also identify poorly designed routines, but correcting them is then more difficult. Some examples of how quality measures can identify excessively complex routines in Fortran software are given in [6].

4 Software Tools: Stand-alone Analysis Software

This section lists a number of software tools, most of which are commercial, which can be used to detect faults, report on standard conformance, measure software complexity and/or restructure Fortran software. A few of these are actually only applicable to Fortran 77 software. They are included because they offer some features which are not common in the Fortran 90 tools and may be useful in analysis of legacy code. In some cases an indication of cost is given, though many vendors do not put this information on their websites.

Also included here are some tools for the conversion of Fortran 77 to Fortran 90. These can help improve software quality through increased readability and maintainability.

4.1 Polyhedron plusFORT

Vendor

Polyhedron Software
Linden House
93 High Street
Standlake
Witney, OX8 7RH, UK
Website: http://www.polyhedron.co.uk

Description

The plusFORT product consists of a set of tools for Fortran analysis checking. The tools currently available in Version 6 are:

plusFORT SPAG. As well as F66 to F77 conversions, this tool is able to convert F77 code to use some F90 constructs, such as DO-WHILE, ENDDO, CYCLE, EXIT, etc. SPAG can also remove dead code and unused variables as well as code "polishing" and standardisation of declarations. The intent of all arguments can also be added to the reformatted code.

Global symbol tables can be produced for further analysis, using GXCHK (see next section). Complexity metrics can be determined for each subprogram. The measures reported are: (1) logical content (number of decision points, comparable to McCabe), (2) all control jumps, where control is transfered to another point, and (3) unstructured jumps. The restructuring of SPAG can often reduce the last two measures.

plusFORT Static Analysis. A full global static analysis can be performed using GXCHK. This uses the symbol tables which are generated by SPAG. It provides checking for variables used before set, argument mismatch, common block problems, etc. It can also generate interface specifications for each subprogram. Call trees and charts showing where variables are used may be produced.

plusFORT Dynamic Analysis Dynamic analysis is made possible by using SPAG to add run time checking statements to the code. Used before set errors, including tests on array elements, are checked in the modified code. The instrumented code is run in the normal way and produces a log file of any faults.

plusFORT Coverage Analysis. The program CVRANAL can be used to instrument Fortran software for coverage analysis. A report is produced as to how often each block has been

executed in the test runs. A cumulative count of the times each block is executed can be made over several runs. CPU time profiling can also be performed.

plusFORT Automake. This automatically checks for dependencies via INCLUDE and USE statements to rebuild an executable file. It is Fortran 90 aware.

Cost

The plusFORT set of tools costs between £1400 and £6990, depending on platform and number of users. A 40% educational or non-commercial discount is available.

Fortran 90 Features

While many Fortran 90 features are mentioned in the documentation, the emphasis is on Fortran 77 code. The documents do not discuss the use of MODULEs, TYPEs or POINTERs. The introduction to the software describes it a converter to structured Fortran 77 with Fortran 90 extensions. It does, however, support F90 source form, declarations and operators.

4.2   Leiden University FORCHECK

Vendor

Forcheck Group
Leiden University
P.O. Box 9604
2300 RC Leiden
Netherlands
Website: `http://www.medfac.leidenuniv.nl/forchek`

Description

FORCHECK supports Fortran 77, Fortran 90 and High Performance Fortran. It also supports many popular extensions. The tool provides:

- Call tree generation;

- Cross reference tables for variables, subprograms, common blocks, etc.;

- Source code listing with error and warning messages inserted.

- Inter-procedure argument checks;

- Static checks for variables used before set, unused variables, etc.;

- A summary report of the code analysis;

A wide range of compilers (and most of their extensions) are supported. These include Cray, IBM, DEC, SGI and SUN Fortran 90 compilers. Several Fortran 95 compilers, such as those from Lahey and Salford, are also supported. The software runs on many different Unix systems as well as DOS and Windows NT/95. A graphical user interface is available on some systems and a command line interface on all systems.

Cost

The software runs on many systems and prices range from 432 Euros on a DOS-based PC to 5864 Euros for a site license. A 40% academic discount is available.

4.3 IPT Corporation Fortran90-Lint

Vendor

IPT Corporation,
1076 East Meadow Circle,
Palo Alto,
California, 94303, USA.
Website: `http://www.iptweb.com`

Description

Details of IPT Fortran 90 can be obtained from the web site. FORTRAN-lint includes a source code analyser that can detect a wide range of potential problems, including:

- Inappropriate arguments passed to functions;

- Inconsistencies in common block declarations;

- Non-portable code;

- Type usage conflicts between different subprograms;

- Unused functions, subroutines, and variables;

- Variables which are referenced but not set.

FORTRAN-lint can be used to:

- Check source files before they are compiled;

- Map out unfamiliar programs. This would include call graph generation and cross referencing;

- Enforce programming standards.

As well as standard Fortran 90, the tool also supports many of the vendor extensions to Fortran 77 and 90. In particular, there is support for for DEC, Cray, MIPS and EPC F90 compiler extensions. The tool can also check details of HPF "comments", if requested to do so.

To make identification of important errors easier, selected warnings can be suppressed. A graphical interface is also included (under X-windows) which can be used to browse the source along with the warning messages and call tree.

4.4 NAG Fortran 90 Tools

Vendor

NAG Ltd,
Wilkinson House,
Jordan Hill Road,
Oxford, OX2 8DR, UK.
Website: http://www.nag.co.uk

Description

In the past NAG has provided software tools for analysis and transformation of Fortran 77 software. They offer a similar set of tools for Fortran 90. These can be run through a GUI which allows easy selection of the source files and operations to perform on them.

The individual tools available include the following.

Polish – for "pretty printing" source code. This includes consistent indentation of loops and conditional statements. Keywords and variables can be individually mapped to upper or lower case to enhance readability. A graphical interface allows simple editing of the formatting options.

Declaration standardisation – this splits up the declarations into groups for easier understanding. Clear comment sections separate scalar arguments from array arguments, local scalars, etc.

Name changer – to alter variable and subprogram names. This allows variable names to be changed to more meaningful strings. It offers a safer conversion process than global edits which might change unintended substrings.

Dependency analysis – generates `makefiles` for a set of Fortran 90 files. With the addition of modules to the language, combined with the existing problem of "include" files, it is a complex task to include all file dependencies in a `makefile`. This can lead to difficult-to-find problems when out of date modules are used by mistake. This tool generates a full set of dependency rules from the selected Fortran files.

Call graph generator – this generates a traditional call tree for a given set of Fortran 90 files, as an aid to documentation and understanding of the software.

Interface builder – this creates explicit interface blocks for Fortran routines which currently only use implicit ones. This enables the Fortran 90 compiler to do much more argument checking of procedure calls. A new module file containing the interface blocks can be generated and appropriate USE statements are inserted in the calling routines. The ONLY qualifier is added to the interface block USE so that it is clear which routine(s) have been defined. In the case of calls to library routines, which are not defined in the code, interface blocks can still be generated, though the user must check that these are indeed correct.

USE statement annotator – this can add appropriate ONLY clauses to USE statements. This makes the way data is imported more explicit to anyone reading the code. It also reduces the chances of subsequent updates to the software using module variables by mistake.

A similar set of NAG tools is available for Fortran 77 software. These include a restructuring tool, similar in operation to SPAG, which attempts to replace old style coding with more readable structures such IF-THEN-ELSE blocks. With the two sets of Fortran tools comes a converter from Fortran 77 to Fortran 90. This includes a graphical user interface and takes advantage of the new free format and declaration style of Fortran 90. The translation tool does not currently try to convert common blocks to modules or map loops onto the new vector notation, as done by Vast77to90. The documentation indicates that a tool for converting commons to modules is under development, along with restructuring and cross reference tools for Fortran 90.

These tools should provide a good path for transforming old Fortran software to Fortran 90 using a well structured and easy to read format. The addition of explicit interfaces to all subprogram calls should significantly increase the confidence in the quality of the software, as it allows immediate detection of linkage errors. The addition of restructuring and common conversion tools should enhance the usefulness of this product.

4.5   Cobalt Blue - FOR_STRUCT, FOR_STUDY

Vendor

Cobalt Blue Inc,
11585 Jones Bridge Road
Suite 420-306
Alpharetta,
Georgia 30005, USA
Website: `http://www.cobalt-blue.com`

Description

Cobalt Blue have two products which can be used for Fortran software analysis. There is some support for Fortran 90, but most of the documentation refers to Fortran 77. The tools available are:

FOR_STRUCT. This product is mainly described in terms of it capabilities to restructure old Fortran 66 code, removing spaghetti code and replacing VAX extensions with standard Fortran 77. However, it also has some support for Fortran 90, including SELECT CASE and relational operators. A demo version is available, limited to 100 lines of code.

The extent of Fortran 90 support in the demo version is very limited. While it accepts SELECT statements and also generates ENDDO for loops, it rejects all MODULE statements.

FOR_STUDY. This tool provides analysis of Fortran software. it includes global argument checking, common block checks and various other potential problems. Call graphs may also be generated. The output reports of FOR_STUDY can be generated in HTML format.

Though there is explicit mention of Fortran 90 feature support, it seems that these does not extend to many import new features such as modules. This makes it of limited use in new Fortran 90 development work.

4.6 N.A. Software Loft90, FortranPlus

N.A. Software Ltd
Roscoe House
62 Roscoe Street
Liverpool L1 9DW
Website: http://www.nasoftware.co.uk

Description

N.A. Software have in the past marketed a conversion tool (F77 to F90), and currently sell a Fortran 95 compiler which includes some global analysis and error checking. These tools are:

Loft90. This product was used to convert Fortran 77 code to Fortran 90 format and to take advantage of the new and safer features of the more recent standard. However, N.A. Software have stopped development of this product and no longer actively sell it.

FortranPlus This is a Fortran 95 compiler for Sparc, Intel and Power PC systems. It supports a number of useful features including interface checking, even when implicit interfaces have been used, and generation of cross reference listings of all variables. A graphical debugger is also part of the system and a program development system keeps track of which files have been recompiled and the options used. New features such as IEEE arithmetic and exceptions and the ISO varying string module are supported.

4.7 Pacific-Sierra Research VAST and DEEP

Vendor

Pacific-Sierra Research
1400 Key Boulevard,
Suite 700.
Arlington,
VA 22209-2369, USA.
Website: http://www.psrw.com

VAST tools

Pacific-Sierra Research sell several software tools under the title VAST. These include translators such as VAST/77to90 and VAST/77toHPF, compiler systems such as VAST/f90 and VAST-HPF and the code optimiser and parallelisation systems VAST/Parallel, VAST/Superscalar and VAST/toOpenMP. While much of the emphasis of these products is on extraction of the maximum performance of existing serial codes on parallel and vector architectures, some can also be useful from the point of view of software quality. The translation tools like VAST/77to90 allow old software to be transformed into the newer format with enhanced readability and maintainability. The optimisation tools could also be useful in transforming existing serial codes of known quality into higher performance versions without the usual level of risks associated with extensive "hand coded" optimisation.

VAST/77to90 Capabilities

This software tool provides translation of Fortran 77 software to Fortran 90. In addition to the relatively simple conversion of fixed form to the new free form, a number of more complex operations can be performed including:

- Addition of INTENT statements to subprogram arguments;

- Removal of obsolete features such as arithmetic IF, etc.;

- Elimination or reduction in the number of GOTO's. In the same way that SPAG restructures Fortran 77, VAST/77to90 attempts to simplify code. This is useful in understanding old software and making future maintenance easier. Unnecessary labels are also removed from DO loops;

- Use of vector notation where possible. Many Fortran 77 loops over vector and matrix subscripts can be reduced to the new and compact array notation available in Fortran 90. As well as often making the code easier to read, this can help optimising compilers identify potentially parallel operations. The tool has options for conservative or aggressive transformation to array notation. In the latter case the code may be harder to read but offer more potential opportunities for parallelism;

- Creation of modules to replace common blocks. Modules offer a safer mechanism for sharing data and VAST/77to90 can automatically generate separate module files. Commons that are in "include" files are also correctly converted;

- Automatic generation of interface blocks. Using explicit interfaces for all subroutine and function calls allows the Fortran compiler to catch errors in argument lists. As well as creating the necessary interface details in a module, appropriate USE statements can be inserted into each calling subprogram;

- Fortran "lint" type diagnostics are generated during the conversion process. This includes warnings about variables used but never set, unused variables and unused code.

Under Windows NT, VAST/77to90 includes a graphical user interface which makes all the processing options easily available for user customisation. In other versions all the options are set via command line switches.

Using VAST/77to90

An older version of this software tool (V1.02X) was available to the authors on an IBM AIX system. It has been successfully used to translate a semiconductor device modelling program from Fortran 77 to Fortran 90. The conversion process was quite straightforward.

A simple example of the translated code was shown in Section 3.3. The ability of this version of Vast to restructure code to remove unnecessary GOTOs was not as good as that of the plusFORT version of SPAG. On two complex Fortran 66 style subroutines, SPAG reduced the GOTO counts from 149 and 67 to 74 and 30 respectively. Vast V1.02X could only restructure these to use 133 and 53 GOTOs. The current version of VAST/77to90 may well be improved in this area.

The Fortran 77 semiconductor code has been developed over a number of years by different people, but has been ported to a range of systems. In the past FTNCHEK and various compiler cross checking tools have been used on it so it was not expected that any significant errors should

be caught. A fault that was found was one where a variable used as an array subscript had been declared double precision instead of integer. In these tests interface blocks were not generated for the code. These would have allowed fuller linkage checking, but may lead to problems where array sections have been passed as arguments in non-standard ways.

The transformed code was tested using the Sun f90 (V1.2) compiler and the DEC OSF/1 f90 compiler (V1.3). Both compiled and ran the transformed code without problems, giving the same answers as the original Fortran 77 version. Performance was unchanged on the DEC machine, not unexpectedly as aggressive optimisation (at the expensive of readability) was not requested. The performance of the Sun f90 version was inferior to the original f77 version, but this is due to the poor performance of this compiler version which has been replaced by V2.0.

DEEP

Pacific-Sierra Research also market DEEP (Development Environment for Parallel Programming). This is a type of Integrated Development Environment (IDE) for parallel programs in Fortran (77, 90, 95) and C. It includes various analysis and debugging tools to help understand the behaviour of parallel codes. While this tool is mainly aimed at parallel optimisation, it also includes call graph display, source code browsing and debugging features useful in software quality assurance.

Cost

VAST/77to90 now comes with a metered license, with cost based on the number of lines converted. For up to 2500 lines the US cost is $125 (May 1999). A 10% academic discount is available. VAST/77toHPF is licensed in a similar way, starting at $250 for 2500 lines.

4.8 Fujitsu Visual Analyzer

Vendor

Lahey Computer Systems,
865 Tahoe Blvd.
Box 6091
Incline Village
NV 89450 USA
Website: http://www.lahey.com

Description

Lahey have recently released the Fujitsu Visual Analyzer. This product provides flow graph analysis for Fortran 95 (and C) software. It can also provide global tracking of common and module usage. An error checking option allows global analysis to detect static errors. Conformance to the Fortran 95 standard is also checked.

This product is only available for Windows NT/95, though it does include a graphical user interface. The single user license is $395 in the USA. The product is included with LF95/Pro compiler.

4.9 Simulog FORESYS

Vendor

Simulog,
1 rue James Joule,
78286 Guyancourt Cedex,
France.
Website: `http://www.simulog.fr/foresys`

Description

The FORESYS product (FORtran Engineering SYStem) is a set of five tools that cover analysis, inspection, restructuring, parallelisation and interfacing to CASE tools. The individual components are:

ADMIN – a Fortran 77 analyser. This tool provides global checking of subprogram interfaces and common blocks. The use of un-initialised variables and sections of unused code are reported. It also creates a database of the analysed program for use by other tools.

IDCM – a tool to examine Fortran source code. This allows the user to browse the Fortran 77 source code, highlighting errors detected by the ADMIN tool. It also allows interactive generation of restructured source in either Fortran 77 or Fortran 90 forms using FOREST.

FOREST – a restructuring tool, including Fortran 77 to 90 conversion. This allows code to be restructured to either Fortran 77 of Fortran 90. Conversion includes use of new declarations, SELECT-CASE, etc. Explicit interfaces can be generated in a separate module. Argument INTENT statements can also be added.

PARTITA – a parallelisation tool. This includes analysis of data flow within existing Fortran 77 code to generate Fortran 90 (or 77) code with good parallel and vector performance.

FORESYS/REV – an interface to the Teamwork/SD CASE tool. This allows high level structural analysis of the source code, including generation of call graphs, etc.

All these tools can make use of a graphical user interface. Though they can be used to convert old code to the new standard, the documentation seems to indicate that the analysis tool can only be used on Fortran 77 code. Supported systems include Sun Solaris, SGI IRIX, IBM AIX, HP-UX and DEC Alpha OSF/1.

4.10 LDRA - Testbed

Vendor

LDRA Ltd,
131 Mount Pleasant,
Liverpool, L3 5TF, UK.
Website: `http://www.ldra.com`

Description

The LDRA software tool Testbed is available on a wide range of platforms and for many programming languages including Fortran. The two main areas of quality that Testbed addresses are static analysis and coverage analysis. However, LDRA have no plans for a Fortran 90 version, so it is included here only for comparison with Fortran 90 tools.

Static analysis

The functions provided by the static analysis tool include:

- Adherence to language and user defined programming standard;

- Level of structures programming used;

- Complexity metrics such as McCabe;

- Static data flow analysis and procedure interface analysis;

- Code re-formatting and variable cross referencing.

Coverage analysis

After static analysis, Testbed can be used to insert instrumenting statements into the software. The modified version of the software can then be run on standard test cases to find which paths through the software have not been tried out. Detailed reports on how often each block of code is executed can be produced.

4.11 Programming Research Ltd – QA Fortran

Vendor

Programming Research Ltd,
Glenbrook House
1/11 Molesey Road
Hersham, Surrey KT12 4RN, UK
Website: http://www.prqa.co.uk

Description

QA Fortran provides a deep flow static analyser to detect errors and inconsistencies. It also provides many measures which can be used to monitor software quality. The software checks for adherence to local programming standards. Fortran 90 is not supported by QA Fortran, and they do not have plans to add it at the present time.

4.12 Other QA tools

There are a number of QA tools providing similar functionality to QA Fortran and LDRA Testbed for Fortran 77 software, such as *Logiscope* (from Verilog, France), *Analysis of Complexity*

Tool (from McCabe and Associates, USA), and *MALPAS* (from TACS, UK). A lot of these are listed in the Appendix of this report. However, as for the above two products, we are unaware of any of these which have yet added full Fortran 90 support.

5 Fortran 90 Compiler Support for Analysis and Error Detection

Many of the global error checking capabilities that in the past have only been available in separate software tools are now being built into commercial compilers. This process is assisted by some of the changes to Fortran, in particular the move towards explicit interfaces to subprograms. Where the user employs such interfaces, along with argument INTENT specifications, the compiler can be expected to catch a whole set of potential errors in argument lists and their use. Some compilers offer more detailed global analysis including warning about variables that are unused, ones used before set and similar problems. Generation of call trees and cross reference maps of symbol names may also be provided.

Run time checks are another important area where many compilers can provide additional checking beyond that of any static analysis tool. Array bound checking is perhaps the most important example, supported by most compilers. Variables that are used before set, particularly in arrays, can also be fully checked by some compilers with run time debugger support. The addition of dynamic memory features to the language, has made the ability to detect potential memory leaks in a code important. This type of check is supported by some run time systems.

Polyhedron have produced a chart comparing a range of Fortran 77 and 90 compilers which is available on their Web site: http://www.polyhedron.co.uk. These results are exclusively for compilers that run on Windows NT/95. A selection of results comparing the debugger and Interactive Development Environment (IDE) support of some compilers are listed in Table 1.

IDE/Debugger Feature	APF	DVF	LF90	LF95	FTN77	FTN95
Fully Integrated Editor	No[1]	Yes	Yes	Yes	No[1]	No[1]
Fully Integrated Profiler	Yes[2]	No[3]	No	No	Yes	Yes
F90 Module Compatible Build[4]	No	Yes	Yes	Yes	No	No
Debug DLLs called from VB/Delphi	No	Yes	Yes	Yes	No	No
Value on Hover	Yes[5]	Yes	Yes[6]	Yes[6]	Yes	Yes
In Scope Variable Window	Yes	Yes	Yes	Yes	Yes	Yes
Display Array Sections	Yes	Yes	Yes	Yes	Yes	Yes
Display PARAMETER values	No	No	Yes	Yes	Yes	Yes
Break on Access	Yes	No	No	No	Yes	Yes
Break on Change	Yes	Yes	Yes	Yes	Yes	Yes
Conditional Break	Yes	Yes	Yes	Yes	Yes	Yes
Change Value	Yes	Yes	Yes	Yes	Yes	Yes
Evaluate	Yes	Yes	No	No	Yes	Yes
Call Stack	Yes	Yes	No	No	Yes	Yes
Step Over	Yes	Yes	Yes	Yes	Yes	Yes
Step Out of	Yes	Yes	No	No	No	No
Run to	Yes	Yes	Yes	Yes	Yes	Yes

Table 1: The comparison of IDEs for win32 Fortran compilers, as produced by Polyhedron. ©Polyhedron Software Ltd. (1998). APF is the Absoft Pro Fortran 90 compiler, V6.0. DVF is the Digital Visual Fortran 90 compiler, V6.0. LF90 and LF95 are the Lahey Fortran 90 (V4.50) and Fortran 95 compilers (V5.50). FTN77 and FTN90 (V1.50) are the Salford Fortran compilers. The Fortran 90 compiler from Edinburgh Portable Compilers (EPS) makes use of the Microsoft Developer Studio and debugger. Notes: (1) Editor and Debugger are separate applications. (2) Subprogram level profiler. (3) External Profiler can be started from IDE. (4) Build should compile minimal set of source files, in the right order, taking account of F90 module dependencies. (5) Double-click on variable - then hover. (6) Right-click on variable.

Polyhedron have have also published some results comparing the ability of PC Fortran compilers to detect common programming errors. A total of 38 tests have currently been collected covering

areas such as argument mismatch, array bound errors, use of uninitialised variables, various illegal assignments and operations on deallocated pointers. The full details and source code for these tests is available on the Polyhedron website. Table 2 just gives the percentage of these errors that each compiler managed to detect correctly.

Compiler	APF77	APF90	DVF	EPC	G77	LF90	LF95	FTN77	FTN95	WAT
Passes	22%	11%	61%	63%	15%	50%	79%	96%	97%	37%

Table 2: The percentage of errors correctly diagnosed on the Polyhedron test set for Fortran 90 and Fortran 77 Windows based compilers. The Fortran 77 compliers pass mark is based on a subset of the tests. The compilers are as described in Table 1 plus EPC, Edinburgh Portable Compilers epcf90 V2.04, G77, Gnu g77 V0.5.22, and WAT, Watcom F77 V10.5.

The Polyhedron results show that the Salford compilers continue to provide excellent error detection. The Lahey, Edinburgh and Digital (Compaq) compilers are also quite effective, though not yet up to the Salford level.

There are many Fortran 90 compilers available on Unix-based computers in addition to those offered by the main vendors for their own platforms. It is not possible to review all of these products in this report, but a short description will be given of some selected ones to highlight the type of compile and run time support which can be available.

5.1   Sun Fortran 90 V1.2

The SUN Fortran 90 compiler is based on the Cray/SGI compiler. The 1.2 version does not have a good reputation for performance but the latest version (V2.0) is claimed to be much improved in this and other areas. We have not used the new version and so all the following comments only apply to the V1.2 compiler.

In common with SUN's f77 product, the f90 compiler supports global checking of subroutine interfaces with the -Xlist option. This creates listing files for each source code file with error and warning messages inserted. The global analysis can be performed even if source files are compiled independently. Cross reference tables for all variables and subroutines are also produced with this option.

One major deficiency of this compiler is the lack of array bound checks. This should be addressed by the new version. The SUN source code browser, which allows interactive examination of call graphs and program flow, appears not to work with f90 V1.2, though V2.0 should add this.

The SUN debugger (dbx) can be used to catch variables that are used before set at run time, though not for variables in common or module storage. It can also detect some illegal memory accesses such as those arising from array bound errors, its ability only partly compensating for the lack of run time checking in V1.2. Another important feature of the debugger is to report on memory leaks. As well as reporting memory that is allocated and not freed, it reports the line at which the allocation occurred.

Using Polyhedron's set of diagnostic problems with the Sun f90 compiler (V1.2) and the dbx debugger, a total of 18 out of 38 errors were detected, about 47%. For comparison, the Sun f77 compiler (V4.2) correctly reported 20 out of 27 errors, a 74% success rate. Much of the difference is due to the lack of array bound checking in f90 (V1.2).

5.2 SGI f90

The SGI Fortran 90 compiler, not surprisingly, offers a very similar global analysis capability to the SUN f90. In this case the analysis is performed by two separate tools, `ftnlist` and `ftnlint`, though in both cases the f90 compiler has to be run first.

The SGI compiler does provide full array bound checking as an option. It also can be used to detect use of unassigned variables at run time via a compiler flag. The standard debugger, `dbx`, does not appear to support memory leak checks. However, there is a new debugging environment called `cvd` which includes many tools, one one of which checks for memory leaks. This did not appear to work with f90 on one system we had access to, but this was without the latest version of the compiler.

The SGI f90 compiler (V7.20) identified 55% of the errors in the Polyhedron diagnostic test set.

5.3 DEC FUSE f90

The Digital (Alpha Unix) Fortran 90 compiler can be used with the support environment called FUSE. This integrates make and source code control programs together the debugger and other tools such as a call graph browser. The call graph browser allows interactive examination of software structures. A cross reference tool is also available, again offering interactive capabilities to examine where and how variables are used and updated. The DEC f90 compiler (V1.3) correctly reported 47% of the errors in the Polyhedron test set.

The FUSE program Visualizer is another tool which can be used to produce diagrams of program structure. It can also take output from other tools such as the Heap Analyzer, which can be used to identify memory errors.

5.4 IBM xlf90

The IBM xlf90 compiler supports Fortran 90 and 95 as well as some OpenMP directives. A HPF compiler is also available from IBM for AIX systems.

Xlf90 includes most of the standard options for error detection and analysis, including array bound checks and variable cross referencing. It can also perform global checking of implicit procedure interfaces and common block usage. Various options are supported to initialise both variables and pointers to enable detection of used before set errors. The xlf90 compiler detected 53% of the errors in the Polyhedron diagnostic set, using an older version of the compiler (V3.2). IBM sells the *VisualAge* system to support software development in a number of languages, but Fortran 90 is not currently mentioned.

More information on threads programming using this compiler is provided in a separate report [4].

5.5 Fortran 90 subset compilers: F and ELF90

These two compilers have been developed to accept only a limited subset of the Fortran 90 language. They essentially ignore most of the old free form Fortran 77 syntax and require the programmer use new features, such as modules, explicit interfaces and subprogram argument intent statements. By forcing the use of the new and safer features, these languages should help improve the quality of newly developed software.

The F compiler is produced by Imagine Inc. and full details can be found on their Web site, `http://www.imagine.com`.

The Elf90 compiler is available from Lahey, `http://www.lahey.com`.

6 Acknowledgements

References

[1] L. Hatton, *The T Experiments: Errors in scientific software*, IEEE Comp. Sci. & Eng., **4**, No. 2, (April-June 1997), 27.

[2] B.A. Nejmeh, *NPATH: A measure of execution path complexity and its applications*, Comm. ACM., **31**, No.2, (1988), 188-200.

[3] T.J. McCabe *A complexity measure*, IEEE Trans. Soft. Eng., **SE-2**, No. 4, (1976), 308-320.

[4] R.J. Allan & C.J. Müller, *SMP Programming*, Technical Report, (CLRC Daresbury Laboratory, 1999).

[5] *Re-engineering Technical Report, Volume 2*, US Air Force Software Technology Support Center (1995) `http://stsc.hil.af.mil/RENG/REENGv2.html`.

[6] T.R. Hopkins, *Is the quality of numerical subroutine code improving?* in "Modern Software Tools for Scientific Computing", E. Arge, A.M. Bruaset and H.P. Langtangen (eds.) (Birkhäuser, 1997) ISBN 0-8176-3974-8.

A Appendix: Fortran Tools listed by STSC

In this appendix we list some data extracted from the report on software re-engineering tools produced by the US Air force Software Technology Support Center (STSC) [1]. The full report is available on the web [5].

The STSC report contains data gathered up to 1995, and hence most of the products are related to Fortran 77 rather than Fortran 90. However, some of these products do now support Fortran 90 and in future more may do so. The ones which are known to support the new standard are discussed in more detail in the main text. Some vendors, such as those of QA Fortran, have indicated that they have no current plans to develop Fortran 90 versions of their products. This is due to limited demand in relation to that for C and C++ tools.

This survey was conducted in the USA and hence most contact addresses are for US vendors, even when the developers may be European. We have just extracted those tools which both treat Fortran and provide functions which may be useful in developing quality software in this language, as opposed to translating it into another language for example. The full report also gives indications of the cost of these products in the USA, which for commercial use is often in the range $2000 to $20000.

Vendor: +1 Software Engineering

Address: *2510-G Las Posas Rd., Ste. 438 Camarillo, CA 93011*

Product: Tree4(Fortran)

Description: Tree4(Fortran) reads in existing Fortran source code and generates the TreeSoft files needed for modelling the existing code. It displays hierarchical representation of the code and allows the user to enhance and maintain the existing source code.

Vendor: Accelr8 Technology Corp.

Address: *5530 S.W. Kelly Portland, OR 97201*

Product: Open ACCLIM8

Description: Automated FORTRAN porting tool for moving VAX/VMS FORTRAN applications onto UNIX systems. Its purpose is to reduce time and cost of moving FORTRAN applications.

Vendor: Advanced Software Automation, Inc.

Address: *3130A Coronado Drive Santa Clara, CA 95054*

Product: ASA 20/20

Description: ASA 20/20 is a member of the Hindsight product family. It allows developers to identify potential problem areas early in the development process.

---

[1]STSC, OO-ALC/TISEC, 278 4th Street, Hill Air Force Base, UT 84056-5205, USA.

Product: AutoAnalyzer

Description: AutoAnalyzer provides a testing and software maintenance environment for software engineers. Builds an interactive structure chart of source code, traces use of global variables, measures and displays test coverage analysis, performance information.

Product: AutoDiagrammer

Description: AutoDiagrammer is straight from source code, generates a low-level detailed procedural logic diagram and a high-level program tree diagram – AutoDiagrammer also will link these together.

Product: AutoStructureChart

Description: AutoStructureChart provides an interactive development environment for software re-engineering and maintenance; it generates a full and organised hierarchical structure chart directly from source code.

Product: Code Generator

Description: The Code Generator includes the following features: language independence, user understandable commands and rules, and syntax checking, and is menu or command driven and user friendly.

Product: Structure & Logic Analysis Module

Description: Structure & Logic Analysis Module generates a hierarchical structure chart directly from source code. This chart helps you understand the calling relationships between parent and child functions. From this high level structure display, you can open diagrams and trace the logic flow.

Vendor: Advanced Systems Technology Corp.

Address: *2200 Defense Highway.,Suite 203, Crofton, MD 21114-2404*

Product: Revengg

Description: Revengg abstracts the structure and interactions of ill-documented programming code into the CaMERA dictionary. Code is interpreted from entry into a specification language and dictionary system for possible re-engineering.

Vendor: Advanced Technology Intl. Corp.

Address: *1501 Broadway Avenue, Suite 1314, New York, NY 10036*

Product: superCASE (SCI)

Description: SuperCASE is an integrated environment supporting top-level design to maintenance. It creates design abstractions from code and facilitates ongoing maintenance and enhancement. It is a reverse engineering tool that extracts design information.

Vendor: ASTA, Inc.

Address: *1 Chestnut St., STe. 205-206, Nashua, NH 03060*

Product: QA FORTRAN

Description: QA FORTRAN is a toolset for analysing, improving, and maintaining FORTRAN programs. It checks for hundreds of reliability, portability, and maintenance issues and computes 28 quality metrics about the code.

Vendor: Cadre Technologies, Inc.

Address: *2880 Lakeside Dr., STe. 231, Santa Clara, CA 95054*

Product: Teamwork/FORTRAN

Description: This product automatically creates structure charts using the Teamwork/SD graphical editor from FORTRAN source libraries. The charts can be cut and pasted into other systems for reuse. The code can be browsed from the structure chart.

Product: Teamwork/IM

Description: Teamwork/IM provides the capability to draw, edit, and check entity relationship diagrams. It is tightly integrated with the data dictionary within the Teamwork environment, and used for database design.

Vendor: Cobalt Blue, Inc.

Address: *875 Old Roswell Rd., Ste. D-400, Roswell, GA 30076*

Product: FOR_STRUCT

Description: Transforms spaghetti FORTRAN-IV and FORTRAN-77 into structured code. Five structuring levels are offered with dozens of output style options. Preserves the original code logic and can remove dead code. Dead code is removed.

Product: FOR_STUDY

Description: FOR_STUDY is a FORTRAN code analysis tool to detect hidden bugs, locate excess code and to generate valuable programmer's documentation for clear and thorough understanding of FORTRAN programs.

Vendor: Digital Equipment Corp. (Now part of Compaq).

Address: *146 Mairr St., Maynard, MA 01754-2571*

Product: DEC FUSE

Description: The DEC FUSE provides code analysis, forward/reverse engineering, re-documentation, and restructuring of several languages. It also provides calling trees and cross reference diagrams.

Product: VAXset/Program Design Facility (VAXset/PDF)

Description: VAXset contains software engineering tools for the development, testing, and maintenance of application programs. The six components include, SCA, LSE, CMS, MMS, PCA, and DTM. VAXset with Program Design Facility also supports re-engineering.

Vendor: Excel Software

Address: *P.O. Box 1414, Marshalltown, IA 50158*

Product: Translator

Description: The Translator automates the generation of diagrams within MacAnalyst and MacDesigner CASE tools from existing source code from many computers written in C, C++, Pascal, Basic, or Fortran.

Vendor: GPP Ges fur Prozessrechnerprogrammierung

Address: *Kolpingring 18a, Oberhaching, D-8024, Germany*

Product: RE-DOC

Description: RE-DOC will create a variety of different, predominantly graphical documents from the original source code for an overview of the software's interrelationships automatically.

Vendor: Iconix Software Engineering, Inc.

Address: *9300 Lee Hwy., Fairfax, VA 22031-1207*

Product: PowerTools

Description: PowerTools consists of seven programs: FreeFlow, FastTask, AdaFlow, ASCII Bridge, SmartChart, PowerPDL, CoCoPro.

Vendor: International Logic Corp.

Address: *100 Pine St., Ste. 2760, San Francisco, CA 94111*

Product: F-SCAN

Description: F-SCAN is a static analyser of FORTRAN source programs that provides call trees, cross references, and analysis of code interfaces.

Vendor: IPT Corp.

Address: *1076 E. Meadow Circle, Palo Alto, CA 94303*

Product: FORTRAN-lint

Description: FORTRAN-lint is a static analyser that detects coding problems similar to what lint utilities do for C programs. The tool does parameter checking and type checking of variables. Other checks include the use of variables before declaration and non-use.

Vendor: Leverage Technologists

Address: *P O Box 4638, Rockville, MD 20849-4638*

Product: FORTRANgen

Description: FORTRANgen is a suite of four tools for revitalising legacy FORTRAN software. It provides a graphical representation which displays system components, relationships, and design elements in a single consistent style.

Vendor: McCabe & Assoc., Inc.

Address: *5501 Twin Knolls Rd., Suite 111, Columbia, MD 21045*

Product: Battlemap Analysis Tool (BAT)

Description: BAT is a reverse engineering and maintenance tool that analyses source code at the system level and computes the McCabe essential complexity metric. Monitors the effect that changes to modules and insertion of new modules have on a system.

Vendor: Meta Systems for LBMS

Address: *3025 Boardwalk Dr., Suite 140, Ann Arbor, MI 48018-1771*

Product: Reveng

Description: Reveng provides automated reverse engineering by analysing individual programs and producing reports on their structure, data structures, and data usage, and by analysing code and modeling whole systems as related objects.

Vendor: MicroTools

Address: *P O Box 2745, Santa Clara, CA 95055-2745*

Product: FORTRAN 77 Programmer's Assistant III

Description: FORTRAN conversion tool and FORTRAN to C/C++ translator. Allows conversion of FORTRAN programs to fully-structured GOTO-less code.

Vendor: NASA Johnson Space Center

Address: *Mail Code PT4, Houston, TX 77058*

Product: REengineering APplications (REAP)

Description: The REAP provides significant maintenance capabilities for FORTRAN programs and support for reengineering. It includes a set of proven tools of FORTRAN analysis, maintenance, and documentation, and a common user interface.

Vendor: OTG Systems, Inc.

Address: *Route 106, Suite 300, PO Box 239, Clifford, PA 18413-0239*

Product: Blue Four

Description: Blue Four structures FORTRAN code. It replaces GOTOs with newer constructs and provides pretty printing.

Product: plusFORT

Description: plusFORT is a multipurpose suite of tools for analysing and improving FORTRAN programs.

Product: SPAG

Description: SPAG is a FORTRAN reconditioning tool. SPAG rejuvenates spaghetti code into structured FORTRAN. SPAG can be customised to support house style FORTRAN pretty printing.

Vendor: QED Software Inc.

Address: *343 South 14th St., San Jose, CA 95112*

Product: Eagle

Description: Eagle provides facilities to deal with changing software at a project level. Is navigation and browsing facilities operate within an editing environment. It also provides the ability to interactively view files, function and data.

Vendor: Quibus Enterprises, Inc.

Address: *19672 Stevens Creek Blvd., Ste 307, Cupertino, CA 95014*

Product: Fortran Development Tools

Description: Fortran Development Tools is a package of utilities for cleaning up Fortran source code and aiding in program development maintenance, and porting. It also produces listings with DOs and IEs graphically highlighted, extract routines from files, and support conditional compilation.

Product: FORWARN

Description: FORWARN is a static analysis tool designed for FORTRAN programs. Using this product improves the reliability, documentation quality, and portability of FORTRAN systems. It furnishes detailed cross reference listings and calling-trees.

Vendor: Reasoning Systems, Inc.

Address: *3260 Hillview Ave., Palo Alto, CA 94304*

Product: Refine/Fortran

Description: Refine/Fortran is an interactive, graphical reengineering tool that lets programmers assess, re-document, quality-check, and convert existing FORTRAN applications.

Product: Software Refinery

Description: Software Refinery is a family of products for building automated software processing tools that analyses and re-documents old code, converts to new languages, checks code against programming standards, and generates test cases from source code.

Vendor: Software AG of North America, Inc.

Address: *11190 Sunrise Valley Dr., Reston, VA 22091*

Product: Predict

Description: PREDICT is an online active data dictionary system that provides integrated operational support for information resource management, software management, and central documentation.

Vendor: Software Business Management Inc.

Address: *234 Littleton Rd., Ste. 2E, Westford, MA 01886*

Product: DecisionVision 1 (DV1)

Description: DecisionVision 1 (DV1) collects and reports measurement information for managing the quality, cost, and schedules of software development projects. It captures code statistics at preset intervals for calculating, graphing and reporting volatility (code changes), reliability and complexity information.

Vendor: Software Systems Design, Inc.

Address: *3627 Padua Ave., Claremont, CA 91711*

Product: FORTRAN Reverse Eng & Document System (FREDoc)

Description: The FREDoc tools give the ability to reverse engineer existing FOR-TRAN code and to understand the design. It helps make sure comment headers are correct and provides two ways of looking at the system-top level or detailed.

Product: GrafBrowse

Description: GrafBrowse is a graphical reverse engineering system and interactive browser for Ada, C, and FORTRAN source codes and design. It analyses codes for certain metrics.

Vendor: Verilog, Inc.

Address: *3010 LBJ Freeway, Suite 900, Dallas, TX 75234*

Product: Logiscope

Description: Logiscope is an automated source code analyser that provides complexity analysis and a dynamic analyser that provide test coverage analysis. Logiscope calculates over 30 different metrics and supports over 35 languages on a variety of platforms. It does graphic test coverage analysis.

Vendor: Vitro Corp.

Address: *14000 Georgia Ave., Silver Spring, MD 20906-2972*

Product: Vitro Automated Structured Testing Tool (VASTT)

Description: VASTT reduces development and maintenance costs, results in higher software quality, and features evaluation of software complexity, testability and maintainability, and evaluation of software risk factors. It is available as a result of the service.

Vendor: Wayne State University

Address: *Department of Computer Science, Detroit, MI 48202*

Product: Visual Interactive FORtran (VIFOR)

Description: VIFOR contains transformation tools for both directions, from code to visual form and from visual form to skeletons of code. This makes it suitable for reverse engineering and for maintenance of existing code.

Vendor: XA Systems Corp.

Address: *983 University Ave., Los Gatos, CA 95030*

Product: DATATEC-DS

Description: The DATATEC-DS is designed to analyse and standardise inconsistent data definitions through a methodology and automated tools. It provides tracing capabilities, impact analysis, formatting and re-formatting capabilities, and more.