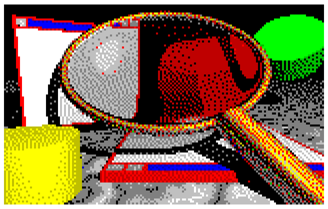
Project Analyzer 4.1



-
Name shadowing
Needless globals report
Library report
-
Problem report
Design quality report
Project summary

Options

Report to
Display
Printer
File...
Printer setup...
Printer font...
-
General options...
Hypertext options...

Add-Ins

Save data for Super Project Analyzer...
Super Project Analyzer
-
Project Printer
Project Graph

List files

Product Analyzer can produce list of files that belong to your project. To get a list, choose **Report|List files**. A dialog box will appear.

You can get lists sorted by:
- File name
- File type

If you want just a list of file names, leave the **Details** box unchecked. If you check it, the list will contain the following additional data:
- Comments
- Which files need procedures in this file
- Which files this file needs

Note: You get the commented lines in the (declarations) section of your .BAS, .FRM, .TXT or .GLB file. The program shows those comment lines that come before the first non-comment line, like this:

```
' This is a function module
' These comments will be reported

Dim Text As String
' This comment will not be reported
```

The file list goes to the output device defined with the **Output goes to** radio buttons.

List procedures

Product Analyzer can produce list of procedures that belong to your project. To get a list, click the **List procedures** button or choose **Report|List procedures**. A dialog box will appear.

You can select if you want the following data included in the procedure list or not:
- Comments
- Which procedures use this procedure
- Which procedures this procedure uses

In the registered version there are also options to list procedures that are or are not needed by your program. This is useful for findingdead procedures.

Note on comments: The program will show those commented lines at the beginning of each sub/function that come before the first empty or non-comment line. Commented lines before and after the sub/function declaration line are shown, as well as procedure description (VB 4.0), if available. An example:

```
' This sub was created by N.N.
SUB MySub (Byval x As Integer)
Attribute VB_Description=Prints x + 5 on the form
' This sub does the following:
' It takes the parameter x and ...

' This line is not shown any more
Form1.Print x + 5

END SUB
```

The procedure list goes to the output device defined with the **Output goes to** radio buttons.

Archive Project Files

Project Analyzer helps backing up your program files. Currently it supports two archiver programs, namely **ARJ** (current version: 2.41a) and **PKZIP** (version at this time: 2.04g). You can also easily build file lists of your project files in order to manually copy/archive them without effort.

Pack Project Files Dialog

Select files to be packed (or files whose file name are to be saved in the list file) in the list box. By default, all VB files belonging to your project are selected, but no DLLs. These are the files that you most probably want to save.

If you want to add files to the list you can click the Add button. Files ending in **.lst** are considered to be list files that contain a list of more files you'd want to add to the list box. You can create these .lst files with any text editor or with Project Analyzer itself.

The **ARJ Command** and **PKZIP Command** fields contain the commands and command switches used to create new archive files.

Sample ARJ Command: ARJ a MYARCH.ARJ !MYPROJ.LST
Sample PKZIP Command: PKZIP MYARCH.ZIP @MYPROJ.LST

You only have to supply the blue part of the command line; Project Analyzer does the rest. Now you can pack by pressing one of the **Pack** buttons.

By default, a temporary file is used to tell the archiver program what files to pack. If you want to save that list file for later use you can click the **Select** button.

Having named a list file, you can save the list file only if you want. In other words, a new button name **List files only** will appear.

Introduction

Project Analyzer is a Finnish shareware tool designed to help creating, maintaining and reporting applications developed in Visual Basic 3.0, 4.0 and 5.0.

Project Analyzer makes a full, two-phase source code analysis. It collects detailed information about all the files, procedures, controls, variables, constants, Types, Enums, classes etc. in the VB project. The results can be viewed on the screen, in textual format in your favourite word processor, as a Help file, in a WWW browser, or on paper.

System requirements

- Windows 95 or NT (or later).
- Visual Basic 5.0 installed, or the run-time libraries. Visit **http://www.aivosto.com/vbcatalog.html** to download the needed DLLs.
- Basically, if you can run Visual Basic, you can run Project Analyzer.

Windows 3.x: If you are using Windows 3.x, get Project Analyzer version 3.1 from **http://www.aivosto.com/vbcatalog.html**. This will run in 16-bit environments.

You only need one version of Project Analyzer for analyzing VB 3.0, 4.0 and 5.0 projects.

Features

Full source code analysis
- File details: procedure and control lists, file level cross-references, lines of code, comment to code ratio and other metrics
- Procedure details: cyclomatic complexity, lines of code, and other metrics, procedure level cross-references
- Find dead and live procedures, variables, constants, Types and Enums. Try the Problem report!
- Variable and constant declaration information, both dead and live
- Call trees
- Project metrics like cyclomatic complexity, comment to code ratio, ...

Reports for maintaining the project
- Sub and Function lists with comments for reporting purposes
- File lists with comments
- DLL reports
- Problem report
- Project summary report
- Variable and constant lists with dead or live variables & constants
- Design quality report
- Needless globals report
- Name shadowing report

Other powerful features
- Hypertext style code viewer in VB Browser
- Project file archiving with PKZIP or ARJ
- Binary extension file viewer (for FRX and other similar files)
- Removing unused constants from CONSTANT.TXT and DATACONS.TXT

See version history for new features.

Why? Why???

Maintain phase costs rule the software industry. When developing applications, especially those that someone else has to maintain, it's important to keep track on what each sub/function does, and what other procedures they need to work.

This is where Project Analyzer can help you. With it you can produce detailed cross-reference reports with supplied comments. You can view call trees. And you can find dead code, dead variables, and dead constants, and various other problems too. You can get the documents on the screen, on paper, or even on your company Intranet as HTML reports.

If you are interested in more examples on how Project Analyzer can help you to optimize your program, see **http://www.aivosto.com/vbtips/vbtips.html**

How to use it?

Using Project Analyzer is simple. Open a **.mak** or **.vbp** file with the Open Project command in the File menu and wait. Project Analyzer will collect information about the **.bas**, **.frm**, **.frx, .cls, .dll** and other VB files in the background.

Note: If you are using VB 3.0, remember to save all your files in text format.

The analysis consist of two phases. The first phase collects the most data, and the second one will check for cross-references and find dead code etc. If a menu command or a button is greyed, don't panic, just wait for the analysis to end.

Project Analyzer does **not** write anything into your code, so it's safe to analyze all your projects.

When the analysis has ended, just double click a procedure or a file to see the results. Remember to push the **Hypertext** button. Don't forget to examine the View and Report menus either, many of the most powerful features reside there (like the Call Tree).

Registration
Collecting information
Terminology
Back to Contents

Collecting Project Information

When you start Project Analyzer, or when you choose **Analyze project** from the **File** menu, you get a dialog box asking you to choose a project file (a VB **.mak** or **.vbp** file). By reading that file, Project Analyzer finds out what files belong to the project. After that, Project Analyzer further looks inside these files.

What kind of files does Project Analyzer analyze?

ProjectAnalyzer examines most file types belonging to your project: .bas, .frm, .frx, .cls, .ctl and other VB source files. As an exception, Project Analyzer doesn't analyze .vbx, .ocx, .tlb, .dsr nor .res files. More about binary files.

If you use VB 3.0, remember to save the files in text format. Project Analyzer can't read VB files saved in binary format, other than the .frx files. If the program finds a binary file it will simply ignore its contents. This will result in incomplete analysis, like showing dead variables that aren't dead in reality. (Project Analyzer can analyze analyze DLL files. It does it based on the declarations in your basic code.)

How does the analysis work?

Project Analyzer reads your files in two phases.

Phase 1/2. Basic information about the structure of your project is collected, including: procedure names, procedure parameter names, variable names, constant names, Type definitions, Enum definitions, control names. Some basic metrics, like lines of code, are calculated too.

Phase 2/2. Cross-references are detected. Project Analyzer scans the whole project to find where procedures are called, variables assigned a value or the values used, constants referenced etc. Some more metrics, like nested conditionals, are calculated too. At the end of Phase 2/2 Project Analyzer calculates additional information based on the cross-reference data: It checks if any procedures, variables, constants, Types or Enums are dead. It also calculates some of the most advanced metrics, like information complexity.

When the analysis is ready, a "Ready!" sign appears at the lower left corner of the program window. Should you want to stop the analysis earlier, you can press the Stop button.

Note: Information is collected in the background. So, when Project Analyzer is examining your files you can view the information the program has collected so far. There are some things you can't do when the collecting is in progress. For example, you don't see if a piece of code is Dead before the end of Phase 2/2, but you see "Wait" instead.

Remember to use Option Explicit! Project Analyzer will get better results if you have declared all your variables. This is good practice anyway.

Terminology
Back to contents

Report Formats

Project Analyzer has very powerful, formatted report options. You can get the reports
a) On the screen
b) On paper
c) On the Clipboard
d) In a file for your favourite word processor
e) In a help file
f) On your company Intranet as HTML files

Use the **Options|Report to** menu command or one of the **Report to** radio buttons to select the report format.
Depending on what you select, you either get a formatted report, or a text-only version.

1. To the display
This is the default. The output goes to a special Display window where you can read the text. You have 2 options to take:
a) A formatted report
b) An unformatted, plain text report (from which you can copy parts to the clipboard)
You can select the type of the report in the **Options|General options** window. More about Options

There are 3 of these Display windows available. You can use them side-by-side.

2. To the printer
If you want, you can choose the output to go to the selected Windows default printer. This report is formatted.

To select a printer, click **Options|Printer setup**. To choose a printer font, click **Options|Printer font**.

3. To a file
Project Analyzer can direct its output to:
a) a Windows text file (plain text)
d) a Rich Text Format (RTF) file (formatted report)
e) an RTF file ready for Help Compiler to turn the report into a Help file (formatted report)
f) an Internet Hypertext (HTM) file (formatted report)

You can even tell Project Analyzer to run an editor, browser or the Help Compiler to open or process the report after it has been created. To do this, just check the **Run browser/editor after report** check box in the Save report dialog box.

Running a browser/editor/Help Compiler

Project Analyzer needs to know which program to run after creating a report. By default, the Windows default associations are used. Press the right arrow button in the Save report dialog box to see which program will be launched.

Possible choices for editors/browsers/Help Compiler:
a) TXT: Windows Notepad or any other text editor
b) RTF: Microsoft Word or Windows WordPad, or any other word processor supporting RTF
c) RTF for Help Compiler:
    1. Microsoft Help Compiler (HC.EXE, HC31.EXE, HCP.EXE, comes with VB 3.0 and 4.0, and is in your \VB\HC directory).
    2. Microsoft Help Workshop (HCW.EXE, comes with VB 5.0, and is in the \Tools\Hcw directory of the VB5 CD-ROM).
    Use of the Help Compiler is recommended instead of Help Workshop, because Project Analyzer writes

a .bat file script for the Help Compiler, which will automatically compile the help file.
d) HTML: Netscape or Explorer, or any other WWW browser

You can change the default by writing your own command line, or by associating the file name extension (**.txt**, **.rtf**, or **.htm**) with your browser/editor. In Win 3.x, you do this from **winfile.exe | File menu | Associate**, and in Win 95, from **Control Panel | View menu | Options | File types**.

Multiple reports in one file

You can add multiple reports in one file by selecting an existing file. If you have selected RTF for Help Compiler, Project Analyzer will also create a Table of Contents page that shows all the reports in that help file.

Note: When producing large reports (over 32kB in size) you can get an "Out of string space" error when the output is going to the unformatted display window. The report will be truncated to 32 kB. The solution is to use any other report type that allows reports of any length.

About Project Analyzer

Contact information - Future versions - Registration

Contact information

Project Analyzer was written by Tuomas Salste, a Visual Basic programmer since 1993. It is distributed by a software company called **Aivosto Oy**, which is owned by Tuomas Salste. Send all email about Project Analyzer to **vbshop@aivosto.com**.

You may also want to visit the VBShop WWW page **http://www.aivosto.com/vb.html** Product information will be available through that page. You will also find information about other Visual Basic tools there.

New versions

Project Analyzer is constantly under further development. New versions with further enhancements can be expected in the future. A registration is valid for one major version number without any need to upgrade. So registered users of version 4.1 will get version 4.2 for free, for example, but not version 5.0. Special low upgrade prices apply.

Distribution of the free updates is done solely on the Internet, by WWW or FTP. **Getting free upgrade versions requires Internet access.**

More about pricing

Shareware registration

Project Analyzer is **shareware.** Simply put, shareware means 'try before you buy'. The unregistered version of Project Analyzer is freely distributable. You can use it to evaluate Project Analyzer to see how it works. You may evaluate it for as long as you like. You can also give it to your friends.

The unregistered version is restricted in some ways. To really use Project Analyzer, you'll need to register. When you pay the registration fee, you get keywords for the full, registered version of Project Analyzer with more features.

Pricing for Project Analyzer

How do I register?

Be sure the read both of the following:
1. The general registration instructions in REGISTRA.HLP. They apply to all Visual Basic tools distributed by Aivosto Oy, also to Project Analyzer.
2. The special registration instructions below. They apply especially to Project Analyzer.

Payment method

The possible payment methods are described in REGISTRA.HLP. All major credit cards, U.S. checks, cash, and some other methods are accepted.

Registration keyword

After registering, you will get a registration keyword that will enable the registered features. Input this keyword in the About box when Project Analyzer starts, and you will be registered.

The same applies to the add-ins for Project Analyzer, but you input the keyword when the add-in starts.

Note: Project Analyzer is an Internet-based product. You don't see it being sold in stores.**There are no physical deliverables. You will not get the program physically in a box. You will get it via email and WWW/FTP**, if you have access to the Internet. In case you don't have an email address, you will receive a letter with the keywords (no diskette).

List variables and constants

Product Analyzer can produce list of the global and module-level variables and constants that belong to your project. To get a list, choose **Report|List variables and constants** in the main window, or press the **Report** button in the Variables and constants window. A dialog box will appear.

You can get two kinds of lists:
1. List all **global** variables and constants in alphabetical order
2. List all **global** and **module level** variables and constants ordered by file they appear in

The file list goes to the output device defined with the Output goes to radio buttons.

Note 1: Variables and constants information is available when analysis phase 1 is complete.

Note 2: You can get more versatile variable and constant lists from the Variables and constants window.

Version History

This version history lists the major changes.
See a more detailed version history in file VERSION.TXT.

Version 4.1 (January 1998):
- Support for compiler directives (#Const, #If...End If)
- The Hypertext, Procedure Details and File Details windows merged into VB Browser
- New add-in, Project Graph
- Project Printer can do Comment manuals
- Comment directives ('$ USED) can be used to bypass problem checking

Version 4.0 (Autumn 1997):
- Added support for Visual Basic 5.0
- User defined Types and Enums included in the analysis
- Call tree and Variables and constants window enhanced

Version 3.1 (Spring 1997):
- Super Project Analyzer
- Project Printer
- Project metrics
- Name shadowing report
- HTML reports
- Help file reports
- Lots of enhancements and bug fixes

Version 3.0 (Summer-Autumn 1996):
- Added support for Visual Basic 4.0
- Class modules supported
- Hypertext window enhanced
- Call tree enhanced
- Needless globals report
- Variables with no type in Problem report
- A lot of enhancements to various features

Version 2.1 (September 1995):
- Project summary report
- Generate constants module command
- Modifications to look better with Windows 95
- Enhancements in the hypertext window
- Average cyclomatic complexity (version 2.1.05 onwards, registered version only)
- About 25% faster analysis (appears in the second phase) due to improving PROJECT.DLL
- Formatted reports on the screen too (version 2.1.07 onwards)
- DLL Report (version 2.1.07 onwards, registered version only)
- Can handle larger projects than the previous versions

Version 2.0 (August 1995):
- The Display window is no more the only way of seeing the results of an analysis. Separate detail windows have been implemented.
- File details and Procedure details can now be viewed on the screen in a separate window in addition to the reports
- File details show the form's icon and controls too
- New Variables and constants window with information about references to them (dead variables & constants, referencing files)
- New Call tree window
- File level call trees added in addition to procedure level call trees
- FRX files can be examined too (just double click the FRX file in the main window)
- Shows declared functions in external DLLs

- Problem report (registered version only)
- Formatted reports using RTF files
- New statistics, like cyclomatic complexity (registered version only) and lines of code for each procedure
- Find command now in the Hypertext and Procedure details window too

Version 1.1 (March 1995):
- Find dead procedures (registered version only)
- Global and module level Variable and constant lists
- Procedural call trees (registered version only)
- Hypertext style code viewer (registered version only)
- Saving the results of the analysis for future use (registered version only)
- A **Find procedure** command to quickly locate a procedure

Version 1.0 (February 1995):
The initial release.

Call tree

(version 1.1 onwards, registered version only)

You can see the procedure and file dependencies in the form of a call tree. The tree is available in three formats:

1. Call tree in a separate window

Use the **View|Call tree** menu command to see an expandable view of the dependencies. The call tree types in this window are:
1.1. Module calls. This tree shows those other modules that are needed for a module to run. A module may use procedures, variables, constants, and also Type and Enum declarations coded in other modules. It may also call another Form's controls and properties.
1.2. Procedure calls. This tree shows the procedure calls inside your project.

The buttons

Pictures	Toggles pictures on/off.
Expand	Expands the call tree from the cursor onwards.
Browse	Shows the selected piece of code in VB Browser.
Report	Turns the tree into a textual report.

Max call depth at the bottom of the window shows how deep the call tree has been at a maximum. This number will increase as you expand the tree.

If a procedure calls itself or another procedure that then eventually calls the first procedure back, it will be marked **<recursive call>** in the tree.

2. Textual report

The reports are available using the **Report|Procedure call tree** and **Report|Module call tree** menu commands. These call trees are the same as on the Call tree window. Three subtypes of a procedure call tree report are available:
2.1. All procedures
2.2. Procedures in selected file only
2.3. The selected procedure only

3. Project Graph

The Project Graph add-in shows graphical call trees.

Note 1: These call tree features are available in the registered version only. In the unregistered version you can see the call tree only for small projects.

Note 2: The call trees are available after analysis phase 2 has completed.

Note 3: You can see dependencies in hypertext or list form too.

Note 4: The program may hang if your call trees are extra large. In this case, see smaller call trees at a time. It is recommended not to take the All procedures call tree on anything but small projects.

Variables and constants window

Project Analyzer can show a list of all global and module-level variables/constants either as a textual report or on the screen. To see the screen list, either press the **Vars & consts** button in the main window, or select the **View|Variable and constant list** menu command.

Options

- Constants
- Variables

- Global vars/consts
- Module-level vars/consts

- Live definitions (those that are used somewhere in the program)
- Dead definitions (those that are not needed for your program). More info

Buttons

Declaration	Opens VB Browser with the (declarations) section where the the selected variable/constant is defined, highlighting it
References	Shows a list of all references and assignments to the selected variable/constant. If it's dead, you the reference list is empty. You can also create a reference report in that window.
Report	Creates a report of all variables/constants that are currently on the list.

Note 1: Variables and constants information is available when analysis phase 1 is complete. References and "dead" or "live" status are available when phase 2 is complete. To refresh the window, close and reopen it after the analysis has completed.

Note 2: To see procedure-level variables or constants, use VB Browser.

Problem report

(version 2.0 onwards, registered version only)

Project Analyzer can provide you with a **Problem report** about your project. It lists some minor but sometimes annoying things.

- Basic files saved as binary - save as text to work with Project Analyzer **and** for better reliability with VB.
- Files without Option Explicit. It is good programming practice to always declare your variables.
- Dead things
- Variables without a specified type (implicit variants)
- Too complex procedures
- Minimizable forms without icon
- Forms with ControlBox but without icon (needed in Win95)
- EXE file without title and icon (this happens if you haven't made an EXE file yet)

It is recommended that you take the Problem report only after the analysis has been fully completed.

Project metrics

(version 3.1 onwards, registered version only)

To monitor your programming performance, programmers often use some simple metric, like lines of code or EXE file size. However, this is not always enough. Project Analyzer helps you to monitor the understandability, complexity, and reusability of your code.

You can see various metrics in many different ways
- **View|Module metrics** and **View|Procedure metrics** windows are good to find long and complex procedures and modules
- Project summary report is good to see the overall performance
- The Design quality report is also good to see the overall performance
- VB Browser shows metrics by module and procedure

Below you can read a brief discussion on understandability, complexity and reliability, and the various metrics used to estimate them.

Understandability

Bad understandability will most probably result in more errors and maintaining problems. You can estimate the understandability of your program with the following metrics:

Lines of code / procedure (View|Metrics, Design quality report)
Procedures longer than 50 lines are often too long. An alarm limit could be 100 lines.

Comment to code ratio and **whitespace to code ratio** (Design quality report)
The more comments in your code, the easier it is to read - and understand. Whitespace is also important for legibility.

Length of identifiers (Design quality report)
The longer your variable, procedure etc. names are, the more likely they are to be descriptive.

Cyclomatic complexity (View|Metrics)
Target for less than 10. See below.

Depth of conditional nesting (see View|Metrics)
Target for less than 5. See below.

Complexity

There are many kinds of software complexity:
- Structural complexity relates comes from conditionals and loops, that is, the flowgraph of the program.
- Psychological complexity means how difficult it is to understand a program. This is very much related to structural complexity.
- Informational complexity is about how much data comes into a procedure and how much goes out.
- Mathematical/computational complexity is about how much time and memory it takes to execute an algorithm.

High complexity may result in bad understandability and more errors. Complex procedures also need more time to develop and test.

The best measures for estimating the complexity of a procedure with Project Analyzer are:
- Lines of code
- Cyclomatic complexity
- Informational complexity
- Structural fan-out

These measures deal mostly with structural and informational complexity. See below for more information.

If you're really interested in structural complexity measures, there is a book that makes a thorough mathematical examination of 98 proposed measures for structural intra-modular complexity.
Horst Zuse (1991) Software Complexity. Measures and Methods. Walter de Gruyter. Berlin - New York.

Reusability

Reusability is a magic word in programming. Measures for reusability are
- Reuse ratios reported by Super Project Analyzer (reuse in a group of projects)
- Structural fan-in. If it's high, the procedure/module is called (reused!) many times.
- Number of class modules

What do the different metrics mean?

Note: You can see most of the metrics only after the whole analysis has been completed.

Lines of code

As simple as it may seem, lines of code is quite a good measure of how complex a program is. Project Analyzer calculates lines of code as follows:

Lines of code = Total lines - Commented lines - Empty lines

Lines of code does not include control declarations in a .frm file.

Cyclomatic complexity
(McCabe)

Cyclomatic complexity is a measure of the structural complexity of a procedure. The higher the number, the more complex the procedure, and the harder it is to maintain it. Cyclomatic complexity for VB procedures is calculated as follows:

Cyclomatic complexity = Number of Branches + 1

Branches are caused by IF, SELECT CASE, DO...LOOP and WHILE...WEND statements.

"Normal" values for cyclomatic complexity range from 1 (very simple) to 9 (moderately complex). If cyclomatic complexity is more than 10, you may want to split the procedure. If it's over 20 you can consider it alarming.

Cyclomatic complexity is the minimum number of test cases you must have to execute every statement in your procedure. This is important information for testing. Carefully test procedures with the highest cyclomatic complexity values.

An average cyclomatic complexity, as well as the distribution of complexity, is reported on the Design quality report.

Note: Cyclomatic complexity often gives quite high values for procedures with long SELECT CASE statements.

Nested conditionals

Nested conditionals metric is related to cyclomatic complexity. Whereas cyclomatic complexity deals with the absolute number of branches, nested conditionals is only interested in how deeply nested these branches are.

If you have a procedure with deeply nested conditionals, you should consider splitting it up. Those procedures can be very hard to understand, and they are quite error-prone too.

Nested loops

Nested loops is a very rough estimate of the mathematical complexity of a procedure. The more nested loops there are in a procedure, the more likely it is that those loops take up a significant amout of time to execute.

Structural fan-in/fan-out

(Constantine & Yourdon)

For procedures:

Structural fan-in = the number of procedures that use this procedure

Structural fan-out = the number of procedures this procedure calls

For modules:

Structural fan-in = the number of modules that use variables, constants, or procedures in this module

Structural fan-out = the number of modules whose variables, constants, or procedures this module needs

A high structural fan-in denotes reusable code.

The higher the structural fan-out, the more dependent that procedure is on other procedures, and more complex too.

Informational fan-in/fan-out and informational complexity

(Henry & Kafura)

Lines of code, cyclomatic complexity, or structural fan-out are not perfect in predicting the "real" complexity of a procedure. For example, a procedure may access a number of global variables and be very complex without having to call many other procedures.

Informational fan-in = Procedures called + parameters referenced + global variables referenced
Informational fan-in estimates the information a procedure reads

Informational fan-out = Procedures that call this procedure + [ByRef] parameters assigned to + global variables assigned to
Informational fan-out estimates the information a procedure returns

Combined, these give a new metric: *informational fan-in x fan-out*. This is reportedly good in predicting the effort needed for implementing a procedure, but not so good in predicting complexity. To predict complexity, we need a new metric: informational complexity. It is calculated as follows:

Informational complexity = lines of code x (informational fan-in x informational fan-out)

Rich Text Format

(version 2.0 onwards)

Project Analyzer can produce formatted reports and save them in a Rich Text Format (RTF) file. A number of popular word processors read RTF files, including Microsoft Word and Windows 95 WordPad. Project Analyzer also produces RTF for the Help Compiler to turn reports into **.hlp** files.

More information on report formats

MS Word

A special option for Microsoft Word users is the **Create Microsoft Word style RTF** check box on the **Save Report** dialog box. This option may produce ugly reports for other word processors (for example, for WordPad).

If this option is set, Project Analyzer optimizes the RTF reports for Word. It uses styles like Normal, Heading 1, Heading 2, ... You can easily create a Table of Contents for your report with Word.

RTF for Help Compiler

You can also create an RTF file for the Microsoft Help Compiler. This way, you can turn the reports into Help files. Project Analyzer will create the necessary .prj file and launch Help Compiler automatically.

The **Create Microsoft Word style RTF** checkbox has no effect on RTF for Help Compiler.

Options

On double click show

These options let you specify what happens when you double click the **file list** or the **procedure list** on the main screen.

1) Report produces the standard reports for a file or a procedure
2) Detail window shows file details or procedure details in a separate window
3) Hypertext window opens the hypertext window with the selected procedure or (declarations)

Show reports to Display as

1) Formatted text Default. Reports to Display are formatted.

2) Unformatted text. Plain text reports in a text box. Choose this if you want to select part of the report for Copy & Paste.

More about report formats

Refer to files by

1) File name. Use the file name (without path) to refer to any files in your project.

2) Module name. If a file has a module name, use it to refer to the file. Otherwise, use the file name.

Use Windows default colors

By default, Project Analyzer uses bright background colors in several windows. Check this option if you prefer ordinary gray backgrounds (in fact, the color used is vbButtonFace, that is, the Windows default button face color).

Hypertext options

Features in the registered version

The following features are available only in the registered version of Project Analyzer. Some of these will work in the shareware version, too, provided your project has no more than 10 source files.
- Dead procedure information
- Dead variable, constant, Type and Enum information
- Project metrics
- Call trees
- Problem report
- DLL report
- Name shadowing report
- Needless globals report
- VB Browser with Hypertext window
- Generate constants module command (for VB 3.0)
- Add-ins

Registration information

Add-ins

Project Analyzer has a couple of add-ins that need a registration for Project Analyzer and a separate registration for the add-in.

- Super Project Analyzer is an add-in tool for analyzing groups of applications.
- Project Printer generates source code reports and comment manuals.
- Project Graph builds a graphical call tree

Project summary

(version 2.1 onwards)

To get a summary of your project, use the **Report|Project summary** command. This report includes some statistics about your project. It also shows some Visual Basic limitations, like the global symbol table size of your project, and its maximum size, 64 kB.

Not all of the measures are exact, these approximated measures are marked as *(approx).*

Generate constants module

(version 2.1 onwards, VB 3.0 only, registered version only)

The use of the CONSTANT.TXT and DATACONS.TXT files in a project tends to take up a large amount of memory, because these files include so many **global const** declarations, and only a few of them are needed in one project. Including a large number of dead constants in a project will fill up the global symbol table, as well as decrease the performance of the program.

If you use Project Analyzer, you can solve the problem easily. When you are developing your application, you can include the CONSTANT.TXT and DATACONS.TXT files in your project; you don't need to cut and paste any declarations to another global module. When your project is about to be ready, just analyze it with Project Analyzer, and select **File|Generate constants module**. This command will generate a new module including only those constant declarations that are needed by your application.

After generating a new module, remove the old one (CONSTANT or DATACONS) from your project and add the new one. You're done.

Note 1: This command applies mainly to VB 3.0 projects. VB 4.0 doesn't require you to declare a vast amount of constants, but has predefined constants instead.
Note 2: This command is available only after analysis phase 2 has completed.

Library report

(version 2.1.07 onwards, registered version only)

This report lists all DLL, OCX and VBX files used by the project.

All declared DLL procedures are listed, and if they are not used, they are marked as "dead".

VBX and OCX files included in your project are listed at the end of the report, as well as any libraries that you have checked in the References dialog box (VB 4.0 and later). If you project file (.vbp) includes information on the version of the library, that is shown too.

Super Project Analyzer

(version 3.1 onwards, needs registered Project Analyzer and a separate registration)

Super Project Analyzer is an add-in tool for Project Analyzer. It is registered separately. It also needs the registered Project Analyzer to work.

A Super Project is a group of ordinary Visual Basic projects that share some files. Super Project Analyzer analyzes these project groups and reports which procedures, variables and constants etc. are shared or dead.

Instructions

To use Super Project Analyzer, save a couple of normal .vbp/.mak analyses using the **Tools|Save data for Super Project Analyzer** menu command. This will write a .sud (Super Project Data) file for each VB project. This file includes information on the modules, procedures, variables and constants used by that project.

After saving a few (at least 2) .sud files, start Super Project Analyzer (**Tools|Super Project Analyzer**). Then open the .sud files with the **Project|Add project** command in Super Project Analyzer. You can now see the modules, procedures, variables and constants used by all of these projects.

Note: In the unregistered Project Analyzer, Super Project Analyzer can only handle projects with less than 10 source files. If you have registered Project Analyzer, Super Project Analyzer can handle up to 25 source files per project without separate registration. The registered Super Project Analyzer, of course, has no such limitations.

Features

Select **Shared only** to limit the display to those elements that are included in at least 2 projects. **Shared Only** may also show dead elements that are included in at least 2 project but are not used by them.

Select **Dead only** to see which elements are not used by any project. You may consider removing them.

Note: Super Project Analyzer does not list VBX, OCX, or FRX files. It does list DLLs and DLL procedures.

Reports

To get a report on what you see on the screen, press the **Report** button, or select the **Report|Report what you see** menu command**.**

Select the **Report|Reuse report** menu command to get a report on reuse ratios. The report includes reuse ratios for modules, procedures and variables & constants by project and for the whole super project as well.

Reuse ratio for modules = Shared modules / All modules in project

Reuse ratio for procedures = Shared procedures / All procedures used in project
(thus excluding dead ones)

Reuse ratio for vars&consts = Shared vars&consts / All vars&consts references in project
(thus excluding ones that are completely dead or that are only assigned to)

Saving a Super Project definition

To save a Super Project definition to a .sup file, use the **File|Save Super Project** menu command. You can later open this file to skip opening each project separately.

Note: In addition to the .sup file, you will need to have the saved .sud files in order for the analysis to work. The .sup file only includes the file names of those .sud files.

Registration

To use Super Project Analyzer, you will need to register both Project Analyzer and Super Project Analyzer.

Prices
More about registration

Project Printer

(version 3.1 onwards, needs registered Project Analyzer and a separate registration)

Project Printer is an add-in tool for Project Analyzer. It needs a separate registration. It also needs the registered Project Analyzer to work.

Project Printer is designed to document VB source code. It can
- Report your source code on the printer or any other output type
- Document references
- Generate hyperlinked source documents in a HTML or HLP file
- Generate a comment manual

Project Printer is not just a plain ordinary code printer. It can
- highlight variables, constants, Types, Enums and procedure calls
- overstrike dead variables, constants, Types and Enums
- list cross-reference information
- report various metrics along with the code
- generate Table of Contents
- even generates hyperlinked source code documents (like the Hypertext window in VB Browser)
- take just your source code comments and make a manual

You can start Project Printer by selecting the **Tools|Project Printer** menu command.

Note 1: Project Printer gives best results after the analysis has been fully completed.

Note 2: The unregistered Project Printer can only handle projects with up to 10 source files. The registered Project Printer has no such limitations.

Output types

Project Printer is versatile. In step 2/3, it asks you to select the output type. Do this in the main window in the ordinary way, as would do when printing any other report.

Paper. Ordinary formatted source code printout with Table of Contents. Optionally with metrics and 'called by' list.

Help file. The easy way to browse a project. Project Printer generates a help file that is very much similar to the Hypertext window of Project Analyzer. Calls are hyperlinked. The Help file is always there for you to take a quick look at your project, even on computers without Project Analyzer.

HTML file for Intranet documentation. Calls are hyperlinked. Useful for a group of programmers. Even store your code bank in HTML.

RTF file. Format the report in any way you want with your favourite word processor.

Other types. You can get the report in any other report type Project Analyzer supports, for example, on the screen, but the results are not as usable as in the above types.

Report types

Project Printer can do 2 report types:
1) Source code
2) Comment manual. The Comment manual option reports comments only, formatted using a special syntax. More information

Options

Project Printer has a number of options in step 3/3. Default values are set based upon which report type has been selected when you continue from step 2/3. It's normally wise to use the default values, but sometimes you'll want to change them.

Source style. **Simple** is a fast, plain text report. **Enhanced** parses through the code, highlights identifiers and overstrikes any dead ones. In HTML and HLP, it also sets hyperlinks for references.

Show contents. On paper, generates a Table of Contents with page numbers at the end of the report. In a Help file, it's on a separate page. In other report types, it is at the start of the report.

List global/module-level vars and consts. Check this if you want a list of vars/consts declared in (declarations). The list includes information on where a variable or constant has been referenced or assigned to, or if it's dead.

Show incoming calls. Lists the procedures that call this procedure.

Show outgoing calls. Lists the procedures that this procedure calls.

Show metrics. Shows various programming metrics, like complexity etc.

Group Subs/Functions/Events/Properties. Normally, the procedures are reported in the order they appear in the source code. This option groups them by procedure type.

Registration

To use Project Printer, you will need to register both Project Analyzer and Project Printer.

Prices
More about registration

Pricing of Project Analyzer 4.1

All prices below are in U.S.dollars.
You can also register in your own currency, but please email **vbshop@aivosto.com** for the price.
The prices are subject to change.

Project Analyzer

Licence	**Single user**	**Additional users**	**Site licence**
Project Analyzer 4.x	$90	$45 / user	$495
Upgrade from Project Analyzer 3.x to 4.x	$30	$15 / user	$165

The price includes all Project Analyzer versions with major version number 4. You can expect a major version number change after the next major Visual Basic version has come out. Minor version upgrades can be expected several times a year. There will be a special upgrade price for registered users when new major versions come out.

An upgrade licence is valid if you have registered an earlier version of Project Analyzer that the upgrade is for. You can upgrade to Project Analyzer 4.x from version 3.x.

A *single user licence* is valid for one user only. If more than one user uses Project Analyzer, you will need to register a *multiuser licence*. If lots of programmers in your organization need Project Analyzer, you can consider registering a site licence. A site licence covers a single organization for an area of up to one hundred miles (160 km) in radius. The price for a site licence price is equal to the price of a multiuser licence for 10 users.

If you already have a a single or multiuser licence and want to upgrade for more users, you will get a discount for the price you have already paid. Email **vbshop@aivosto.com** for details.

Add-ins	**Single user**	**Additional users**	**Site licence**
Super Project Analyzer	$30	$15 / user	$165
Project Printer	$40	$20 / user	$220
Project Graph	$30	$15 / user	$165

The above add-ins require a separate registration. In addition, they need the registered Project Analyzer to work. An add-in licence is valid for all future Project Analyzer versions (the possible versions 5, 6, etc).

Source code	
Project Analyzer 4.x source code for VB 5.0 Pro	$198
Source code upgrade from 3.x to 4.x	$50

The source code is also available. This price includes all updates with the major version number 4. You don't have to purchase multiuser or site licences for the source, but you need to have registered Project Analyzer. If you want the source code, you should register all the add-ins too, because the source includes all the add-ins (but **not** the source for any **.ocx** controls, like Lgraph.Ocx).

The source has been developed in Visual Basic 5.0 Professional. There are no special documents for the source except for comments in the code.

The source code licence allows you to use the source for any purpose in your own organization. However, you may not distribute the source in any form. For example, if you have modified the source, you may not sell it to anyone without a separate agreement.

How do I pay?

The possible payment methods are described in REGISTRA.HLP. All major credit cards, U.S. checks, cash, and some other methods are accepted.

Note: **There are no deliverables. You will not get the program physically in a box. You will get it via email and WWW/FTP, if you have access to the Internet.** In case you don't have an email address, you will receive a letter with the keywords.

More about registration
Features in the registered version

Terminology

Procedure	A Sub or Function, either a VB procedure or a DLL declaration.
Dead procedure	A sub/function that is not executed at run-time. It can be 1) a procedure that is not called anywhere in the program 2) a procedure that is only called by other dead procedures
Live procedure	The opposite of a dead procedure. The following kinds of procedures are always considered as live procedures. 1) Event Subs. There is no way to detect at design-time if an event occurs at run-time. 2) Procedures that implement an interface. Procedures in the abstract base class are analyzed normally, but in the child class, procedures like MyBaseClass_MyMethod are considered live, because there is no way to detect if they are actually called at run-time or not. 3) Public members of public classes. Public classes are often called from outside the project they are in, so Project Analyzer can't analyze the calls to them. See Public class below. 4) Procedures defined as used by compiler directives
Dead constant	A constant whose value is not used in the code.
Dead variable	*1) For non-object variables (e.g. Integers, Variants, user-defined Types...):* A variable whose value is not used in the code. - A variable that is assigned to can be dead if its value is not used. - A ByRef procedure parameter that is assigned to is never dead, because its value may be used in the procedure where the call came from. *2) For object variables (e.g. Class, Control, Form variables):* An object variable that is not assigned nor referenced to. - An object that is referenced is never dead - An object variable that is defined as Dim x As New ObjType is never dead - An object that is created with Dim x As ObjType is not dead if it is assigned to, for example, by Set x = y.
Dead Type/Enum	A user-defined Type is dead if it doesn't appear in the code. An Enum is dead if neither its name nor any of its members appear in the code. Public Types and Enums in Public classes are never considered as dead, because they may be used by other projects. (See Public class below)
Public class	A Public class can be one of the following: a) .cls files with Instancing <> Private (ActiveX EXE, DLL or OCX projects) b) .ctl files with Public = True (OCX projects) c) .dob files (ActiveX EXE or ActiveX DLL projects)
Variable assignment	A variable is given a new value with one of the following statements x = 1 Let x = 1 Set x = y

	For x = 1 To 100 For Each x In group
Variable reference	Every occurrence of a variable that is not an assignment. Assignments and references can exist in a single statement, like below: MyObject.MyProperty = 3 This is a reference to MyObject and an assignment to MyProperty.
Object variable	A variable that can hold an object. The object can be, for example,a Class, a Form, or a Control. Variants can also hold objects, but they are classified as non-object variables, because there is no way at design-time to detect if a Variant holds an object at run-time.
Global	An entity that is available throughout the whole project. Example: a variable declared Global or Public in the (declarations) section.
Module-level	An entity that is available inside one module only. Usually declared Private in the (declarations) section.
Procedure-level	An entity that is available inside one procedure only. Examples: procedure parameters and variables declared inside a procedure.

Metrics described

Contents

VB Browser

(version 4.1 onwards, registered version or small projects only)

VB Browser has three distinct features:
1. Hypertext
2. Module details
3. Procedure details

You can display VB Browser by double-clicking the Files or Procedures lists in the main window. You can also select the View|VB Browser menu command, or use the toolbar. The currently selected module or procedure will be shown.

VB Browser works after analysis phase 1/2 has completed, although it gives more results after phase 2/2.

1) Hypertext

The hypertext page is a read-only code window with sophisticated cross-reference features. You can see called procedures and declarations of constants, variables, Types and Enums by clicking underlined words.

By right-clicking a highlighted variable, constant, procedure, Type or Enum you get a menu with cross-reference information. This works with all global and module-level definitions, but not procedure-level ones.

Options
You can change the colors of different types of code by selecting **View|Hypertext options**. If you want, Project Analyzer can **~~overstrike~~** **dead** things too. Overstrike is available in the registered version.

Hint: Project Printer can save your source code in hypertext format into a Windows Help file. The result is quite similar to the Hypertext window, but you no longer need Project Analyzer to view the file. You may, for example, distribute your code in a Help file with hyperlinks.

2) Module details

The Module details page shows information on single files, either normal VB code files or DLL files too.

Basic files

- Size in kilobytes
- File version: An internal version number of the file. This version is not necessarily the same as the version of your VB. Not all file types have a version number.
- Controls (Form files only)
- List of files that use this file (after phase 2/2)
- List of files that this file uses (after phase 2/2)
- Total lines, and lines of comment and white space
- Subs and functions: Name, type (functions only), lines of code, and 'Dead' status if the procedure is not used by your project. Double click a procedure to jump to it.
- Variables, constants, Types and Enums defined in this file: Global and module-level definitions.
- Controls on Form, UserControl and PropertyPage files

DLL files (DLL, EXE, ...)

DLL's are analyzed like normal Basic files, except:
- No variables, constants, procedures, Types or Enums in the DLL are detected
- Files that are called by this DLL are not analyzed
- The lines of code in a DLL cannot be analyzed

- DLL file version is not analyzed
Only those procedures that were declared with a Declare statement are shown.

3) Procedure details

The Procedure details page shows information about a specified sub, function, or property procedure. It also shows information about (declarations) sections and declared DLL procedures.

The following information is shown:
- Procedure declaration
- The 'deadness' of the procedure (after phase 2)
- Total lines and lines of code (doesn't apply to DLL procedures)
- Various metrics (after phase 2)
- Procedures that call this procedure
- Procedures that are called by this procedure (doesn't apply to library procedures)

Some parts of the information will not be available before the analysis phase 2 has been completed. 'Wait...' indicates that the analysis is not complete yet.

Binary Files

Project Analyzer cannot analyze most binary files. Some of the file types that Project Analyzer doesn't analyze are VBX, OCX, TLB, DSR and RES. Moreover, Project Analyzer cannot analyze VB files saved as Binary in VB 3.0.

However, Project Analyzer can analyze DLL files based on Declare statements. It can also analyze FRX files, as well as CTX, PGX, DOX and DSX files which are similar to FRX files in structure. The rest of this topic is about the FRX view that you can display by double-clicking a FRX file (or similar) in the main window of Project Analyzer.

FRX/CTX/PGX/DOX/DSX files

Project Analyzer can examine the contents of a binary extension file that is there if you saved your Form file in text format (VB 3.0 has this option, later versions do it automatically) and the form contained pictures or any other binary data. In Visual Basic 4.0 and earlier, there are only FRX files available. In VB 5.0 and later, there are lots of other files that are essentially of the same type, for example, a CTL file is often accompanied by a CTX file.

These files include the binary properties of your controls. You can see what's inside a binary extension file by double-clicking it in the main window of Project Analyzer, or by selecting File Details.

Project Analyzer cannot show all kinds of binary data, just the most usual types of them. Some supported properties are:
- PictureBox.Picture
- Form.Icon
- Form.MouseIcon
- TextBox.Text (if it's a multiline box)
- ListBox.List and ListBox.ItemData

If Project Analyzer cannot show a picture, it will try to show the data as text.

You can extract a picture from your VB project by saving it to a file. Just click the **Save As** button.

Project Graph

(version 4.1 onwards, needs registered Project Analyzer and a separate registration)

Project Graph is an add-in tool for Project Analyzer. It is registered separately. It also needs the registered Project Analyzer to work.

Project Graph builds graphical call trees. It can build them at both module and procedure level. The call trees Project Graph builds are more detailed than those given by the Call Tree window, because Project Graph shows both
1) Forward calls, originating from the selected module or procedure, and
2) Backward calls, coming from other parts of your project to to the selected module or procedure

Double-click a node to move in the tree. Right-click a node to display a context-sensitive menu.

Expanding. Because call trees easily grow very large, the initial depth of a call tree is limited to 3 levels. However, you can manually expand branches by right-clicking a node where the tree can be further expanded (indicated by an arrow like this: -> 4) and selecting **Expand**. You can also move to a node and press the **x** key to call expand. Sometimes expanding will produce a spaghetti-like call tree. To avoid this, expand nodes in up-to-down order.

Note 1. Project Graph works after the analysis has been fully completed.

Note 2. The unregistered Project Graph can only handle projects with up to 10 source files. The registered Project Graph has no such limitations.

Registration

To use Project Graph, you will need to register both Project Analyzer and Project Graph.

Prices
More about registration
Features in the registered version

Comment manual

(version 4.1.04 onwards)

Comment manual is a feature of the Project Printer add-in. It creates a manual-like report based on comments in your source code.

Comment manual takes all comments immediately before and after your Sub/Function/Property declaration line and makes a report out of them. All comments before the Sub/Function/Property line are taken, as well as all comments after it up to an empty line or a normal code line. This latter rule holds for comments at the start of the (declarations) section too.

Normal comments are reported as such, just comment marks (') and Rem statements are removed. There are 2 syntax rules that you can use to format the output, namely Titles and Parameter tables.

1. Titles

```
' MyTitle: Text line 1
' Text line 2
```

This will produce a title line with text "MyTitle", and below that "Text line 1" and "Text line 2". The following would also give the same result:

```
' MyTitle:
' Text line 1
' Text line 2
```

2. Parameter tables

```
[Param] Descr
```

This will create a little table with 2 columns and default headings, like this:

Parameter	**Description**
Param	Descr

A parameter table ends when there are no more commented lines with brackets.

3. General tables

```
[*3*] [Heading1]   [Heading2]   [Heading3]
[a] [b] c
[d] [e] [f]
```

[*n*] marks the start of a general table with n columns of equal width. The [Headings] after [*n*] create bolded column headings for the table. You can create a table without headings by putting [*n*] alone on a line. A table may contain a maximum of 5 columns. A table ends when there are no more commented lines with brackets.

Note: You may omit the brackets [ ] around the last column if the row has more than one column.

This is what will be created:

Heading1	**Heading2**	**Heading3**
a	b	c
d	e	f

Example

This is an example of a commented procedure:

```
Function PrintText(x As Integer, y As Integer, Text As String) As Boolean
' Prints text on the screen at (x, y)
' [x] The x-coordinate in pixels
' [y] The y-coordinate in pixels
' [Text] The string that will be displayed
' Returns:
' True if successful
' False in case an error occurs
' Remarks: Calls Err.Raise 12345 if Text is too long

(and the real code would start here)
```

This is what you would get:

PrintText

Syntax
Function **PrintText**(*x* As Integer, *y* As Integer, *Text* As String) As Boolean

Prints text on the screen at (x, y)

Parameter	**Description**
x	The x-coordinate in pixels
y	The y-coordinate in pixels
Text	The string that will be displayed

Returns
True if successful
False in case an eror occurs

Remarks
Calls Err.Raise 12345 if Text is too long

Comment directives

Version 4.1.06 onwards

Comment directives are a way to tell Project Analyzer to behave in non-standard ways. You can use them to ignore dead code, for example.

Comment directive syntax

```
    '$ <directive> [ : <directive> [ : <directive> ...]]
```

Comment directives always start with `'$`
The prefix $ marks the start of a comment directive line. Rem $ is not accepted.

You can put multiple directives on one line by separating them with a colon (:).

List of possible directives

1. USED
2. END

USED

Marks things as being in use (and not "dead").

Syntax:
```
    USED [<option>] [BEGIN]
```

<option> is optional, and it can be one of the following:

`PARAM`	Directive affects procedure parameters only
`PROC`	Directive affects procedures only (both VB and DLL procedures)

If you omit <option>, the directive affects all things, namely procedures, variables, constants, Types and Enums.

The optional parameter BEGIN marks the start of a BEGIN...END block (see Scope below for more information).

Examples:

```
    '$ USED
    Sub MyProc ( x )
```

Defines MyProc and x as used

```
    '$ USED PARAM
    Sub MyProc ( x )
```

Defines x as used, but doesn't affect MyProc.

```
    '$ USED PROC
    Sub MyProc ( x )
```

Defines MyProc as used, but doesn't affect x.

END

Ends a BEGIN...END directive block. See Scope below for more information.

Scope

What code do the directives affect? There are 3 alternative formats you can use.

1) Next line

```
'$ USED
Dim MyVariable As Integer
```

A directive on an otherwise empty line affects the immediately following line. In this case, MyVariable is marked as used.

2) Current line

```
Dim MyVariable As Integer '$ USED
```

A directive after a code line affects the current line. In this case, MyVariable is marked as used.

3) Begin...End block

```
'$ USED BEGIN
Dim MyVariable As Integer
Dim Text As String
'$ END
```

The directive affects all code before the next '$ END directive. In this case, both MyVariable and Text are marked as used.

You can nest multiple BEGIN...END blocks.

Terminology
Contents