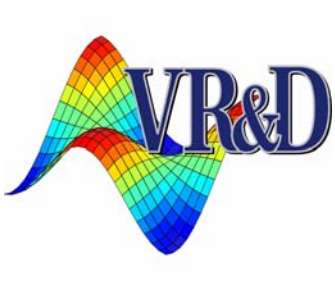


# VisualDOC

## LIST OF NEW FEATURES

***VERSION 7.1***

November 2012


1767 SOUTH 8TH STREET, SUITE 200
COLORADO SPRINGS, CO 80905
Phone: (719) 473-4611 Fax: (719) 473-4638
http://www.vrand.com
Email: visualdoc.support@vrand.com

## COPYRIGHT NOTICE



## WARNING



## DISCLAIMER



## TRADEMARKS MENTIONED IN THIS MANUAL

# VisualDOC 7.1

VisualDOC 7.1 includes a large of major and minor changes which are primarily aimed at improving the user-interface, significantly reducing memory usage, and improving the performance. The following is the list of additions and enhancements in this release of VisualDOC.

# New File Definition UI

The file definition user-interface has been completely rewritten. The snapshot of the new UI is shown in **Figure 1**.

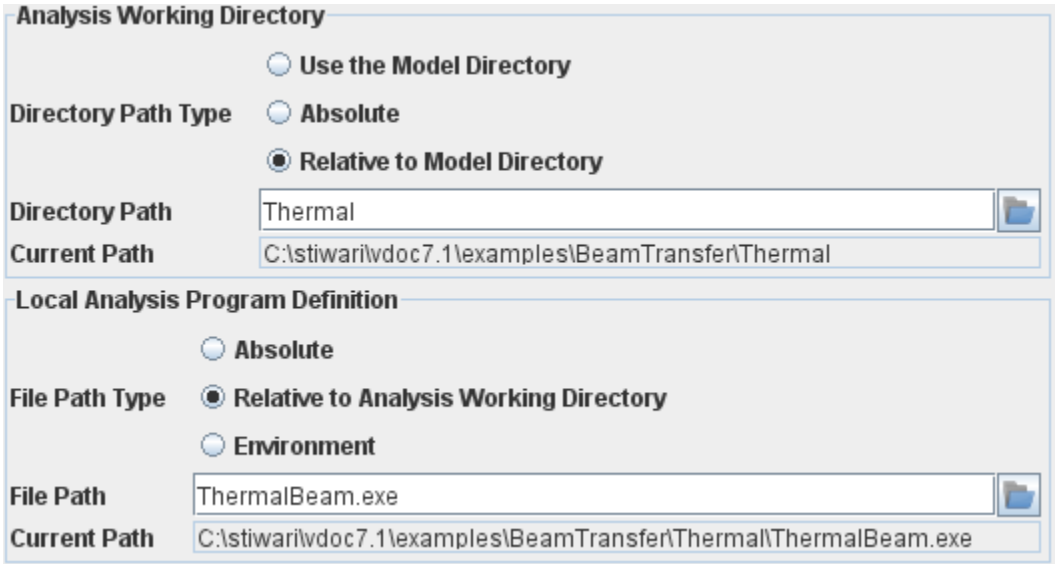


**Figure 1 File Definition UI**

In **Figure 1**, two File Definition UIs are shown. The first UI is used to configure the "Analysis Working Directory" and the second UI is used to configure the "Local Analysis Program Definition". The following are the major enhancements.

1. The same UI is used throughout the VisualDOC to configure the location of a file/folder. This consistency ensures identical behavior throughout the software.
2. The UI now supports additional File Path Type options. The following options are supported.
   2.1. *Absolute Path*: The user-specified path is absolute. This type of path will not change when the VisualDOC model is moved to another location or another computer.
   2.2. *Relative Path*: In this case, the path specified by the user is relative to another reference path. The reference path may be the model directory, remote working directory, or a path specified in another File Definition UI.
   2.3. *Model Directory*: In this case, the path is the model directory itself. The path string always points to the model directory. Moving the model to a different directory/computer will change this path automatically.
   2.4. *Environment*: This option is useful and available only when specifying an executable program. When this option is used, VisualDOC will search the PATH environment variable on the current system to determine the location of executable program.

3. A new text field with name “Current Path” has been added. This field is not user-editable and shows the resolved path string. This is the actual path that will be used during runtime. This path is updated in real-time (instantaneously) during the editing process. This path is automatically updated when the model is moved to a different folder/computer.
4. The file definition UI is now hierarchic; i.e. the path specified in one UI can depend on another UI. For example, in **Figure 1**, the location of the analysis program depends upon the analysis working directory. When the analysis working directory is modified, the location of the analysis program (Current Path field) is automatically updated to reflect the new location. This property is true across all components throughout VisualDOC.
5. The new UI now supports sensible auto-complete/configure. This feature assists the user in configuring the UI. For example, if the user changes the path type from relative to absolute, then the current resolved runtime path becomes the user-specified absolute path. Similarly if an absolute path is changed to relative, then VisualDOC resolves the current path relative to the reference path, and the resolved path is set as the user-specified path name. The new UI automatically resolves and updates many different path change combinations. The user can always override the changes if desired.
6. For remote configuration, the new UI solely relies on String processing to resolve/combine/update the path names. In such a case, the path name String itself is used to infer whether the remote host is Windows or Linux/Unix. If the user-specified path name does not contain sufficient information to infer the remote OS (e.g. if the path name only contains a single file or folder), Unix style path name separator (forward slash “/”) is used for combination.

## New File-Backup Component ( )

A new component to backup files and folder during a VisualDOC simulation is added. This component can be used to specify which files and folders to be backed-up (saved to a user-specified location) during the simulation. It is often desired to backup generated simulation data for later use or post-processing. For example, if an expensive simulation involving LS-Dyna is performed in VisualDOC and the user intends to post-process the LS-Dyna results for different iterations, this component can be used to backup the entire simulation data for each iteration for later usage. The snapshot of the property editor of the File Backup component is shown in **Figure 2**. The following features are supported by this component.

1. It can backup user-specified files as well as folders. The folder contents are backed-up recursively.
2. The “Backup Directory Definition” which specifies the location (root) of the backed-up folder structure is user-configurable and is done using the new File Definition UI.
3. The name of the Backup folder is user-configurable. The user can specify a string that contains pre-defined macros (e.g. <%iter%>) which are replaced by iteration id during runtime to generate backup folder names as the simulation proceeds.
4. A preview functionality is added that generates a preview of the backed-up files. The preview option shows the tree view of the contents of the backup folder structure as they would appear after the backup. A snapshot of the backup folder structure is shown in **Figure 3**.
5. A “Purge Backup Files” option is provided. The purge option can be used to permanently delete backed up files and folders. The purge option shows the user the actual list of files and folders (after checking the hard-disk contents) that would be deleted and requires the user to confirm the delete operation.

6. A flexible and powerful auto-fill option is provided. The auto-fill option automatically populates the list of files that the user may want to backup. Any file that is read/written by a component that belongs to the same sub-flow as the File Backup component is eligible for backup. All the components that are part of the same sub-flow are polled for a list of eligible files which are then added to the backup list. Furthermore, the auto-fill preserves the configuration of the files (whether they are relative, absolute, etc.) while populating the backup list so that if the model is moved to a different folder/computer, the backup works exactly as intended. The user can edit the automatically populated files if desired.

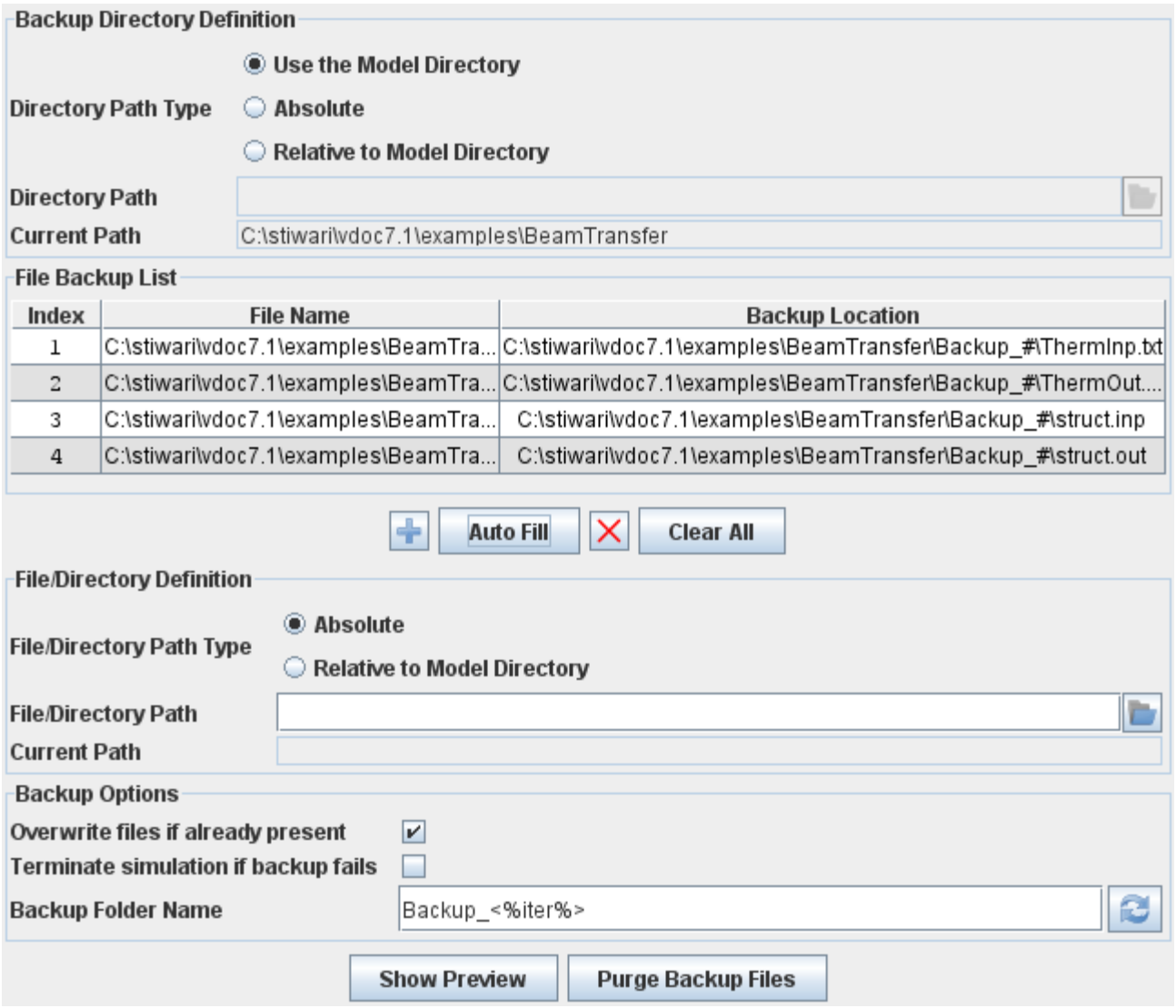


**Figure 2 Property Editor of the File Backup Component**

7. A “Clear All” option is provided that allows the user to clear the list of files to be backed up. The user must confirm this operation.
8. The user can also choose to specify whether the files should be overwritten during the backup operation. The default is to overwrite the files during backup. Whenever a file is overwritten, a warning message is printed to notify the user about the specific file that was overwritten.
9. The user can also choose to terminate the task execution if backup fails for some reason. The default is to print a warning message whenever a file could not be backed up and continue the simulation (task execution).

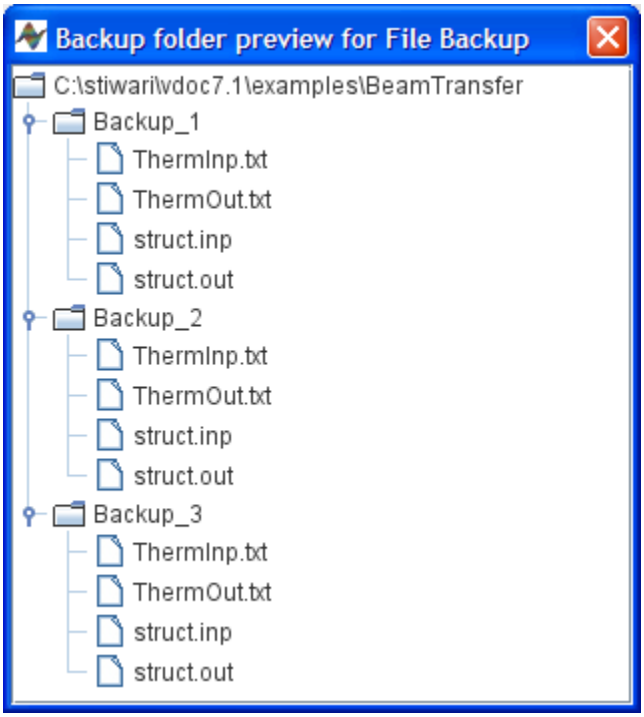


**Figure 3 Preview of the Backup folder structure**

10. The File Backup component uses the new Java I/O functionality to copy files/folders and its performance is similar to the native copy/paste operation of the operating system.

## Redesigned Excel Component

The Excel component has been completely redesigned and rewritten. The redesign is primarily aimed at improving the performance and making it easy to configure the component for reading/writing large amount of data. The list of new/improved functionality in the Excel component is as follows.

1. VisualDOC 7.0 only allowed a single Excel component in the flowchart. This limitation is removed and now the user can add any number of Excel components to the workflow.
2. Multiple Excel components can read/write to the same Excel file if desired. They can also read/write to the same portion (cells) of the Excel file if desired. Concurrent read/write however is not allowed.
3. In VisualDOC 7.0, only real numbers could be read-from/written-to Excel. With this version, the user can also read/write integers and strings. More data types can be added in future depending upon the user request.
4. In version 7.0, the user had to individually specify the cell (sheet number, column name, and row number) for each element even if the cells were contiguous. With version 7.1, the user only needs to specify the sheet number, start column, and start row of the cell. The number of columns and rows to read/write is automatically determined from the size of the data. VisualDOC automatically populates the column/row values for each cell in the contiguous block.
5. A major limitation of the Excel component in version 7.0 was its ability to only read/write single cells at a time. With version 7.1, a contiguous set of cells can be read/written at once. With large data (several thousand elements), dramatic performance improvement can be observed with the new Excel component.

6. For contiguous cells, the user has the option to choose column vector (single column), row vector (single row), or a 2D matrix (more than one column and row). This information is automatically extracted from the data size and dimension.
7. The error checking and reporting functionality has been significantly improved. While reading/writing, VisualDOC can determine if the Excel file is already opened by another process - in such a case, VisualDOC informs the user to close the Excel file and then run the simulation. It also checks if the Excel file can be opened (has the correct read/write permissions), a cell is empty or missing, has the correct runtime type, etc. If a cell (or a set of cells) are incorrectly configured, VisualDOC prints an appropriate error message and also highlights the incorrect cells for easy editing. VisualDOC now also checks for missing data when reading an Excel file.
8. Microsoft Excel from version 1997 to 2010 are supported. Some of the functionality (e.g. independently deciding whether to save changes) is only supported on Excel versions 2003 and later.
9. A maximum of 256 sheets and 702 columns (A, ..., Z, AA, ..., ZZ) are supported. There is no specific limit on the number of rows that can be read/written.
10. As a result of the above changes, for very large data (~100,000 contiguous rows or more), VisualDOC 7.1 is ~1000 times faster than VisualDOC 7.0.

## Memory Efficiency and Performance Improvements

A major focus of this release of VisualDOC is to significantly reduce the memory usage and improve the performance. Many different modules have been redesigned and/or completely rewritten to improve the performance and efficiency of VisualDOC.

## New Compressed Binary File Format

VisualDOC 7.0 models were stored as a key-value pair in ASCII file format. With VisualDOC 7.1, a new compressed binary file format is introduced. The older (deprecated) format is still fully supported, but it is strongly recommended to use the new compressed binary file format which is also the default for version 7.1.

## New Highly Efficient Data Structure for Storing Components and Data

All the components in the VisualDOC flowchart and the user-defined simulation data (inputs, outputs, etc.) now use a new data structure which is similar to version 7.0 but requires significantly less memory to store the same amount of information.

Both the above changes are completely transparent to the user, and the user will not notice any difference to the user interface. As a result of the above and many other modifications, on large problems (data size of 100,000 or more), the following improvement is typical.

- ~70% reduction in heap size to store a model in the memory.
- ~90% reduction in heap size to read a model into the memory
- ~99% reduction in the size of the database file (when the database file contains only the model and no simulation results).
- ~50x faster read performance
- ~30x faster write performance
- ~2x faster undo/redo performance and ~40x less memory usage for undo/redo

- ~10x faster equation component
- ~2x faster all other components

In VisualDOC 7.0, the size of the Vector data was limited to 1000. As a result of the above enhancements, the limit on the size of the Vector data is increased to 100,000. That is, for roughly the same amount of memory usage, the user can now create and use ~100x more data.

## New Raw Array Data Type

VisualDOC 7.1 includes a new data type referred to as Raw Array. This new data type is designed to be used in scenarios where large data sizes of up to few million are desired. The Raw Array data type is similar to the Vector data type but has the following major limitations.

- The Raw Array data type only supports real numbers. It does not support integer, boolean, and strings.
- Design components do not support the Raw Array data type. Raw Array data type can still be added to the design component but a Raw Array data type cannot be a variable or a response (objective, constraint, etc.). This limitation is due to the fact that it is often the analysis that contains huge amount of data and the design components (e.g. optimization) typically involve less than 100,000 variables, objectives, or constraints.
- The user cannot individually configure any property of the individual elements of the Raw Array data. Only the values (initial, current, etc.) can be independently edited. The Raw Array data (handle) itself is fully configurable, but its elements simply inherit the property. For example, the user cannot independently link the elements of Raw Array data, the entire Raw Array data has to be linked at once.
- The Raw Array data is not stored in the database. The components that typically need Raw Array data have input and output files which contain the simulation data that can be backed up by the Backup component. Individually monitoring (e.g. creating a 2D plot) each element of the Raw Array data is not typically desired.

Due to the above limitations, the Raw Array data type is roughly an order of magnitude more efficient than the Vector data type. The user can create data with sizes of up to few million. The actual data size that can be used depends upon the available memory and the specific component. All the control and analysis components fully support the new Raw Array data type.

## Modifications to the User Interface to Handle Large Data

As a result of the improvements to the Vector data and the introduction of the new Raw Array data, several GUI components in VisualDOC have been modified and/or rewritten to handle large data. Most of the GUI elements that have been modified appear identical to that of version 7.0. The following GUI elements in particular have been modified.

- The data tree table UI has been modified to show only a part of the Vector/Raw Array. By default, the data tree-table only shows the first 10 and last 10 elements. The user can configure how many elements to be shown by default. This configuration option is stored as a preference. A partial snapshot of the data tree-table showing only the first and last few elements is shown in **Figure 4**.
- The data extraction module is rewritten to work with the large data. The modified UI can easily handle very large data, few million iterations of an analysis component, and database files which are few GB in size.

| Name | Input/Output | Data Type | Value |
|---|---|---|---|
| thickness_initial | Output | Raw Array | Real |
| thickness_initial[0] | Output | Raw Array | Real |
| thickness_initial[1] | Output | Raw Array | Real |
| thickness_initial[2] | Output | Raw Array | Real |
| thickness_initial[3] | Output | Raw Array | Real |
| thickness_initial[4] | Output | Raw Array | Real |
| thickness_initial[61435] | Output | Raw Array | Real |
| thickness_initial[61436] | Output | Raw Array | Real |
| thickness_initial[61437] | Output | Raw Array | Real |
| thickness_initial[61438] | Output | Raw Array | Real |
| thickness_initial[61439] | Output | Raw Array | Real |
| PSHELL_Num | Output | Raw Array | Real |
| PSHELL_Num[0] | Output | Raw Array | Real |
| PSHELL_Num[1] | Output | Raw Array | Real |
| PSHELL_Num[2] | Output | Raw Array | Real |
| PSHELL_Num[3] | Output | Raw Array | Real |
| PSHELL_Num[4] | Output | Raw Array | Real |
| PSHELL_Num[61435] | Output | Raw Array | Real |
| PSHELL_Num[61436] | Output | Raw Array | Real |
| PSHELL_Num[61437] | Output | Raw Array | Real |

**Figure 4 Data TreeTable**

Raw Array Editor/Viewer for data with name PSHELL_Num

| Index | Name | Initial Value | Current Value |
|---|---|---|---|
| 46516 | PSHELL_Num[46515] | 0.0 | 47448.0 |
| 46517 | PSHELL_Num[46516] | 0.0 | 47449.0 |
| 46518 | PSHELL_Num[46517] | 0.0 | 47450.0 |
| 46519 | PSHELL_Num[46518] | 0.0 | 47451.0 |
| 46520 | PSHELL_Num[46519] | 0.0 | 47452.0 |
| 46521 | PSHELL_Num[46520] | 0.0 | 47453.0 |
| 46522 | PSHELL_Num[46521] | 0.0 | 47454.0 |
| 46523 | PSHELL_Num[46522] | 0.0 | 47455.0 |
| 46524 | PSHELL_Num[46523] | 0.0 | 47456.0 |
| 46525 | PSHELL_Num[46524] | 0.0 | 47457.0 |
| 46526 | PSHELL_Num[46525] | 0.0 | 47458.0 |
| 46527 | PSHELL_Num[46526] | 0.0 | 47459.0 |
| 46528 | PSHELL_Num[46527] | 0.0 | 47460.0 |
| 46529 | PSHELL_Num[46528] | 0.0 | 2909.0 |
| 46530 | PSHELL_Num[46529] | 0.0 | 47461.0 |

**Figure 5 Raw Array Data Editor/Viewer**

- Synthetic data editor has been modified to work with Raw Array data type which could be few million in size.
- A New Raw Array Data Editor/Viewer is added that can be used to edit and examine the values of all the elements of the Raw Array data. This editor is extremely efficient and can handle Raw Array data of any size. A snapshot of the Raw Array data editor/visualizer is shown in **Figure 5**.
- Several other minor modifications have been done throughout to the GUI to more efficiently work with large data.

## Database Optimizations

The VisualDOC database stores all the models as well as the simulation data - all in one file. One of the major functions of the database is the storage and retrieval of simulation results. The simulation results may be retrieved for post processing, generating summary reports, or exporting it to external sources such as text files.

The database engine has been modified to improve the performance as well as reduce the file size. The following are the major changes to the database engine in VisualDOC 7.1.

- In VisualDOC 7.0, simulation data for design components was stored in a single large table. With version 7.1, the simulation data is stored in a large number of small tables. The storage into tables is split to enable localized queries which often fetch data only from a single table.
- With version 7.1, the same amount of simulation data can be stored in roughly half the space as compared to version 7.0.
- Added indexing support for large tables. The user has the option to index the database (by choosing the appropriate menu item) to significantly improve the query times. Indexing the database theoretically reduces the query time from linear to logarithmic.
- With version 7.1, the database is now 100% ACID compliant. It implies that database corruption cannot occur in the event of power loss, operating system crash, or any other error. The contents of the database will always be in a consistent state and the user can always retrieve the contents from the database.
- An option has been added to save simulation data only for designated components. The user can choose to save the simulation data for all the components, no component, only the design components (default), or the designated components.
- An option to purge the entire simulation data for all the tasks and also compact the entire database at once has been added. This option is useful if the user only wants to store the model information in a small file.

Owing to the numerous additions to the database functionality, a new Database menu has been added in version 7.1. Most of the database functions can be accessed from this menu.

## Other Modifications

- A new utility to clear the old VisualDOC log files has been added. The user can choose to delete all the log files or the log files that are more than one week old.
- A menu item to release unused memory has also been added. VisualDOC automatically releases the unused memory at different occasions. With this menu item, the user can trigger this process as and when desired.
- All the bugs discovered since version 7.0 have been fixed in this release.

## Version Compatibility

One of the features of VisualDOC is to be able to communicate with various different software such as Matlab and Excel.Each release of VisualDOC is tested with the earliest versions that it supports and also the latest release of different software products. VisualDOC has been tested with the following software versions.

- Java

  VisualDOC works with both the 32- and 64-bit Java. Java versions from 5.x to the version 7 update 7 are supported. Java 6 or later is recommended. VisualDOC will not run on Java version 4 or earlier.

- Windows

  VisualDOC works on both the 32- and 64-bit Windows. It has been tested on Windows XP, Vista, 7, and pre-release version of Windows 8. On Windows 8, VisualDOC defaults to the classic UI and does not support the new Metro interface.

- Linux

  Only the 64-bit Linux distributions that have kernel version 2.6.x are supported. VisualDOC works with most major distributions of Linux.

- Microsoft Excel

  Excel versions 1997 to 2010 are supported. It is strongly recommended to use Excel 2003 or later with VisualDOC.

- Matlab

  Release 2009a to Release 2012b are supported. Matlab releases prior to 2009a are not supported.

Even though VisualDOC supports older versions of the software that it integrates with, it is strongly recommended to use the latest versions of the software whenever possible. It is especially suggested to use Java 6 or later for performance and efficiency reasons.

## New Examples and Documentation

Five new examples showcasing the new and advanced features of VisualDOC have been added with this release. The new examples are as follows.

1. Stress Ratio: This example demonstrates the use of Raw Array data type.
2. DOT Scaling: This example demonstrates scaling of variables (it addresses the issue of sensitivity)
3. Data interpolation: This example shows how to fit cubic splines to arbitrary one-dimensional data and then use the spline as an approximation to the data.
4. Car-Body optimization: This example presents a case study on performing body-in-white optimization with VisualDOC as the optimizer and GENESIS as the finite element software.
5. RBDO: This example shows how to perform reliability-based design optimization efficiently with VisualDOC.

A new DOT Build Instructions manual has been added with this release. All the accompanying manuals have been updated for the new release.