SAND REPORT

SAND2002-2237
Unlimited Release
Printed July 2002

PDF Requirements Linking

Stephanie M. Castillo
Kent de Jong

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/ordering.htm

SAND 2002-2237
Unlimited Release
Printed July 2002

PDF Requirements Linking

Stephanie M. Castillo
Information Technology and Data Modeling

Kent de Jong
Product Realization Standards and Processes

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1012

Abstract

This report describes the PDF Object Linking Extension (POLE) and how it came about. POLE is an extension of an existing DXL script called Outdoors that provides a linking mechanism to files outside of DOORS. Our modifications expand the script's capabilities to link to bookmarks within PDF documents. PDF linking allows for traceability to be maintained between DOORS objects and the requirements within PDF files.

Intentionally Left Blank

Contents

Figures

Tables

Acronyms

DXL	DOORS Extension Language
OLE	Object Linking and Embedding
PDF	Portable Document Format
POLE	PDF Object Linking Extension

PDF Requirements Linking

It's Easier to Catch Fish With A Pole

Imagine you were just hired by a large fish hatchery. Your job is to monitor thousands of fish and track their growth. You have a map of the complex, which shows what tank has which type of fish. There are 50 to 1000 fish living in each tank. Your job is to locate and catch particular types of fish for analyzing and testing.

One day your boss comes to you with a fish egg and asks you to find the fish that produced the egg. You look at your map and find the tank in which it is located. You proceed to the tank where there are many fish swimming. You begin by catching one fish by hand. It is not the one you are looking for so you throw it back. You continue catching and return one fish at a time. After 2 hours, you catch the right fish, and take it to your boss. He performs 15 minutes of testing and throws it back in the tank with all the other fish.

A week later, your boss again asks you to find the same fish; some of his results were inconclusive and he needed to perform further testing. By this time you have memorized the tanks, so you proceed back to the tank. Once again, one by one, you catch each fish and throw it back until you find the right one. This time, because you already knew what tank it was in, it only took an hour and forty minutes to find. You take it to your boss. He performs a few minutes of testing and again throws the fish back with all the others in the tank.

You decide there has to be a better way to do business. Every day your boss asks to you to find a particular fish, you spend hours trying to catch it for him. He performs a few minutes of testing only to throw it back into the tank.

You decide to tag all the fish. That way you would not have to catch each fish, you would just have to look for the tag and pull out one fish. Yet, you'd like your job to be even easier. Finding a tank and looking through all the fish to find the correct tag leaves you bleary-eyed and tired. You want the fish to come to you.

You remember that a state of the art fishing pole had been introduced back at a fishing convention. This pole can automatically find a blue tail fish by its tag, catch it, and reel it in. Unfortunately, the hatchery only has red tail fish. You take this tool back to your workstation. After several modifications and improvements, it catches blue tail fish and red-tail fish.

Soon your boss asks you to find the baby fish of the mother he had been testing. You reach for your improved fishing pole, enter the appropriate tag name and press a button. The fishing line shoots out and lands in tank A. Fifteen seconds later, the baby fish is in your hand. You proudly take it to your boss and he performs his 15 minutes of testing. When he's finished, you take the baby from him and gladly throw it back into its tank.

Now you may ask, "What does this fish story have to do with requirements management and DOORS®?" Well…

Our Challenge

…Imagine sitting in your office enjoying your morning cup of coffee. Your boss walks in and tells you your new assignment is to manage requirements for a multi-million dollar project. What are you going to do? First, you ask the project's manager about the project's needs for requirements management. The manager gives you over 500 pages of documents and says "Here are the customer requirements, manage these." What would you do?

Our requirements management adventure started with requirements in many paper based documents. We realized that to do what had been done in the past was futile. So, we decided a requirements management tool was in order. The question is, which one? We had plenty of experience with user-hostile tools and knew that only an expert could work with them. Due to the magnitude of the job, for this project to be successful, the engineers developing the system and subsystems design would need to use the requirements management tools. After a big GULP, we began the search.

We explored options with other tools before we serendipitously happened upon DOORS®. We liked DOORS® a lot. It was easy to use. We felt sure that the engineers, even after their previous experiences with requirements management tools, would find it acceptable. We were hooked.

One of the reasons we were attracted to DOORS® was because of its potential to maintain traceability between requirements, specifications, and tests. Tracking requirements in hundreds of pages makes the capability of linking between specific requirements and tests a necessity. The number of requirements we must maintain are in the thousands. They are embedded in text, tables, and graphics, all of which must be related…somehow.

Many of these requirements were on paper or in four or five different software packages. Unfortunately, DOORS® did not read paper and it really did not like some of the older software packages. We decided to convert our paper-based documents and ancient software based requirements to an electronic format that DOORS® could read. We started with one paper based document and began the conversion process and entry into DOORS®.

In addition to our paper- and ancient software-based documents, we also had a significant number of requirements stored in Portable Document Format (PDF) files maintained by our customers. These files, once created would not change materially over the course of several years or decades. The requirements inside these PDF files were items to which we wanted our DOORS® database to provide traceability. Unfortunately, DOORS® would only open a PDF file rather than link directly to a requirements object located inside a PDF file.

We found the work of converting documents to DOORS® to be very challenging and difficult with the cost of one document's conversion and entry into DOORS® nearly $15,000. We decided there must be a better way.

In July of 2001, in the rather hot and muggy south central town of San Antonio, we happened upon a treasure. We met Ron Lewis who as a graduate student created a linking mechanism from DOORS® to Microsoft®'s Word, Excel, and PowerPoint. My colleagues and I breathed a big sigh of relief. We had found a way to deal with our horrendous paper documents…or so we thought. As it turned, OutDoors did not do PDF…So what were we going to do now. Since last August, we have been experimenting with OutDoors and in the process have created a PDF linking mechanism which extends the capabilities of OutDoors to grapple with the requirements which are located PDF files.

Existing Importing And Linking Capabilities of DOORS®

The developers of DOORS® anticipated the need for documents that originated in other file formats to be included in a requirements management system. In the creation of DOORS® they incorporated two methods of including and linking to external documents in a requirements database. The two available options are to import the document or to insert an OLE Object.

When importing an external document into DOORS, the user is limited to only importing documents in Plain Text, Rich Text, Microsoft Project, Spreadsheet, FrameMaker or Interleaf file format. When documents, in any of the six supported importable formats, are imported into DOORS®, parts of the document are stored as individual objects. Links can then be created to and from the individual objects. Importing a document allows a user to maintain traceability between information as necessary. One thing to remember is, a document that is imported into DOORS® must be maintained in DOORS® to preserve the integrity of the data.

An alternative to importing a document, that allows a document to be maintained in its original format, is to insert an OLE Object. An OLE Object allows any type of Windows based document to be linked to from a DOORS® object. Keep in mind that it is a link to the document as a whole. This means, traceability is not maintained between objects in DOORS® and information within the document.

Why The Existing Capabilities Were Not Enough

The reason an extension of the existing capabilities was necessary was that many times, requirements, justification and other additional information that needs to be tracked, would not be located within DOORS®. One reason for not putting a document into DOORS® is it's file format. It is not possible to import a document that is not in one of the six supported formats. Another reason is the size of the document. Some requirements documents are so large and contain so many tables and images that formatting them for proper importation would be a time consuming and expensive effort. A third reason is the desire or necessity to maintain the document in its original file format, not in DOORS®. Given these limitations in our project, we still needed a way to link from DOORS® objects to information within external documents. Linking within a document would eliminate the manual searching process we were trying to get away from by implementing DOORS®.

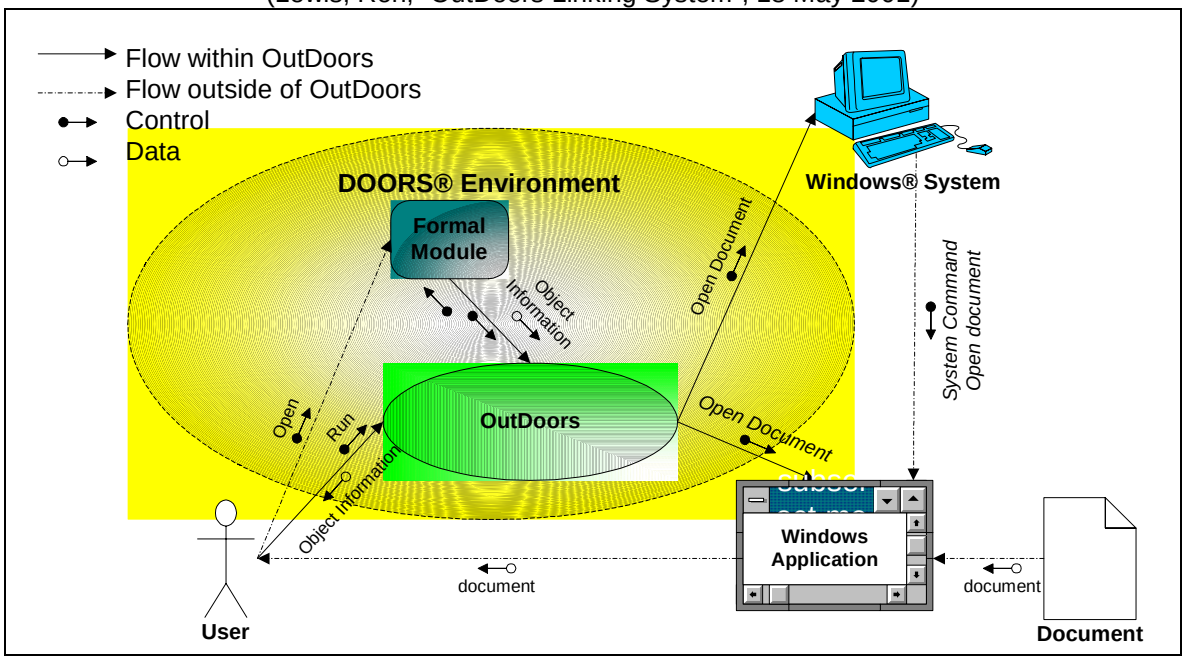
One of the main requirements we had for external linking was the capability to link to text within PDF files. Some of the requirements documents we had access to were only available to us in PDF format. We also wanted the option to store other external documents in PDF format so the information would be protected and easily transmittable.

OutDoors Linking System

We learned about OutDoors at the 2001 DOORS® conference. OutDoors, created by Ron Lewis, is a linking mechanism that is implemented in DXL. It allows users to easily link to and view external documents from objects in a DOORS® module. What was most important to us was it's capability for links to be created to an object within Microsoft® Office files. A link can be made to bookmarks in Microsoft® Word files, cells in Excel files, and slides in PowerPoint files.

A user populates a formal module with objects and an, 'OutDoors', attribute that contains the path of the external link. The user activates OutDoors, from the menu, within the formal module. The module sends OutDoors information on its objects and attributes. Once the user selects the desired object and link, the document is opened in one of two ways. If it is an Microsoft® Word, Excel or PowerPoint, OutDoors sends an OLE command to the application, telling it to open the document and go to the specified part of the document. If it is any other file, OutDoors sends a command to the system telling it to open the application and document. (See Figure 1) The ability to link to parts of documents would save having to manually search through hundreds of pages to find a specific piece of information. The only capability OutDoors was missing for our purposes was a way to link to bookmarks in PDF files. We determined OutDoors could be used as a basis to obtain the functionality we were looking for.

Figure 1: OutDoors Dataflow Diagram
(Lewis, Ron, "OutDoors Linking System", 18 May 2001)

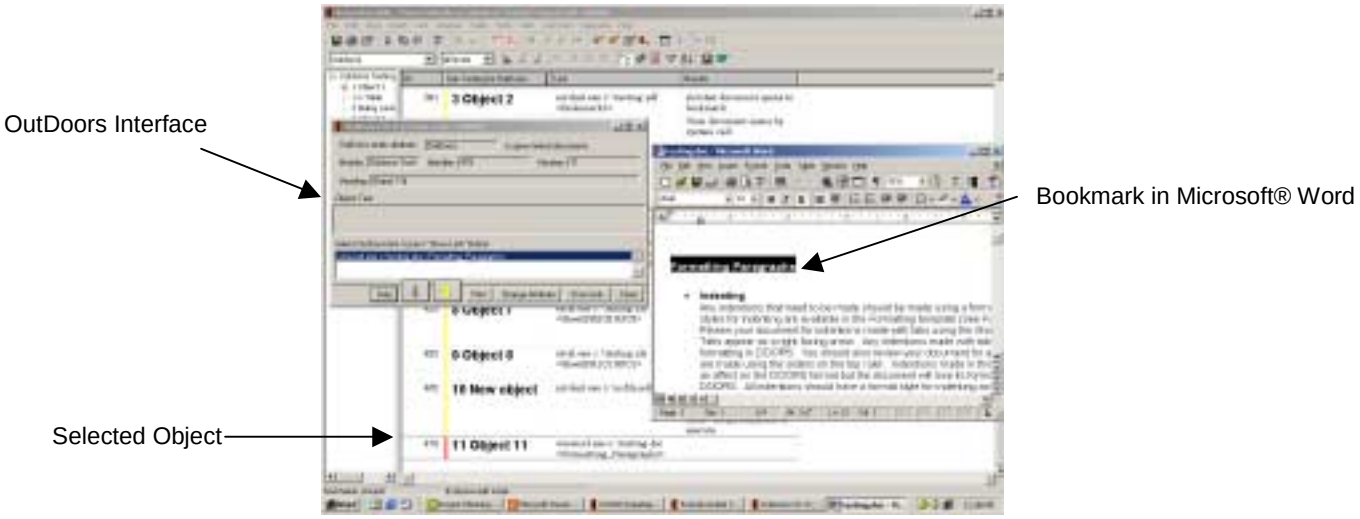


Modifications and Additions To OutDoors

The two underlying technologies of OutDoors are DXL and OLE (Object Linking and Embedding). We made minor modifications to the original DXL to make OutDoors more intuitive and user friendly. Other modifications were made to eliminate some redundancy and to make the code more efficient.

We added on to OutDoors, the functionality we needed. Our additions to OutDoors consisted of both OLE and DXL. We tailored it to support the functionality of linking to bookmarks in PDF files.

Figure 2: OutDoors Linking System Interface



What is OLE?

OLE is a mechanism created by Microsoft that allows users to integrate data from different applications. It is the same technology that is used when an OLE Object is inserted into a module. It is implemented through classes and methods that are incorporated into various programming languages (i.e. Visual Basic, C++, DXL). DXL includes methods with which DOORS® can be used to control other Windows applications. The application in control, DOORS®, is referred to as an automation client and the application it controls is an automation server. The automation server allows the client access to its automation methods. With access to the server's methods, the client application can control the server application. In our case, Adobe® Acrobat® is the automation server (***Acrobat® Reader® does not support OLE automation***). Table 1 shows an example of using OLE methods in DXL to open documents from other applications.

Table 1: Implementation of OLE with DXL

MICROSOFT® WORD	ADOBE® ACROBAT®
//OPEN MICROSOFT® WORD DOCUMENT s="Word.Application" objWord=oleCreateAutoObject(s) put(autoArgs, "filename") oleGet(objWord,"Documents",objDocuments) oleMethod(objDocuments,"Open", autoArgs) olePut(objWord,"visible",true)	**//OPEN ACROBAT APPLICATION** s="AcroExch.App" objAcroApp=oleCreateAutoObject(s) oleMethod(objAcroApp,"Show") **//OPEN DOCUMENT** s="AcroExch.AVDoc" objAVDoc=oleCreateAutoObject(s) put(autoArgs, "filename") put(autoArgs,"") oleMethod(objAVDoc,"Open",autoArgs) clear(autoArgs)

The PDF Object Linking Extension

The result of the additions to OutDoors is the PDF Object Linking Extension. It is a mechanism incorporated into OutDoors that allows links to be created from a DOORS® object to a bookmark within a PDF file. The filename and name of the bookmark are entered into the 'OutDoors' attribute. Once the object is selected and the 'Show Link' button is pressed a call is made to Acrobat commanding it to display it's window. A second command is sent, telling it to open the specified document. A third command is sent, telling it to perform the action of the bookmark, which in our case is the 'Go to View'. The 'Go to View' bookmark action performs much the same function as an Microsoft® Word bookmark. It navigates through the document to the specified destination (See Figure 4). One difference between an Acrobat bookmark and an Microsoft® Word bookmark is the way the bookmark is initialized and viewed. In Microsoft® Word® a bookmark displays selected text. In Acrobat, a bookmark displays a selected view. A view includes the position and magnification of a page. See Figure 3 for a graphical representation of the relationship between OLE, DXL, OutDoors, and the PDF Object Linking Extension.

Figure 3: Relationship Diagram

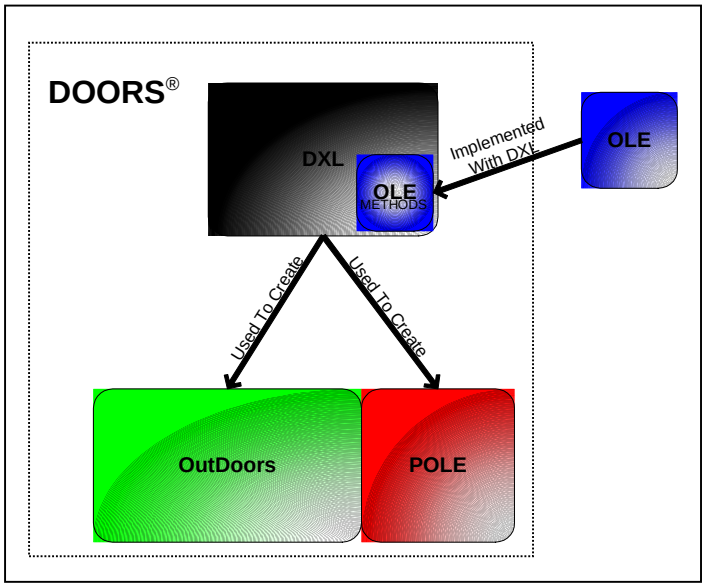
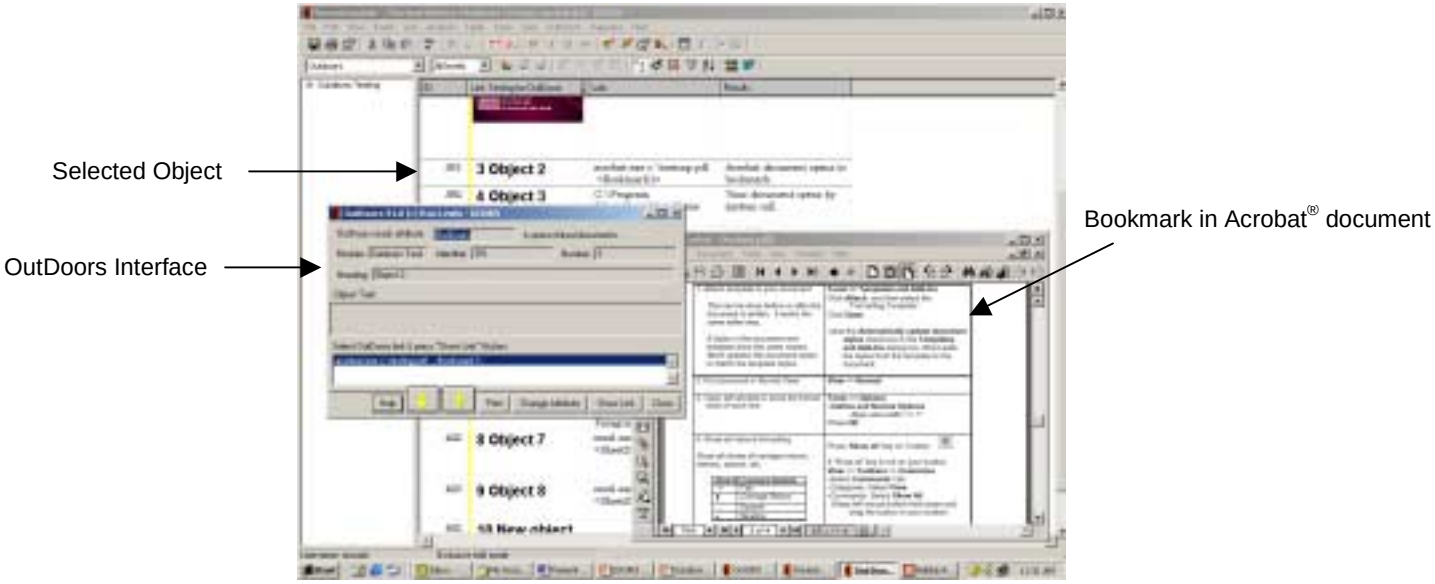


Figure 4: OutDoors & POLE Linking System Interface



Advancements

One of the advancements we would like to see in the future is the ability to link to sentences or individual requirements in a PDF document. Unfortunately, the inability to do so now is a significant limitation and weakness of Adobe® Acrobat®. It does not include a feature to create a bookmark of selected text and display it with a differentiating appearance. The user is limited to creating links to and viewing a portion of a page or paragraph determined by a view. This needed functionality is one that may come with a future release of Acrobat®.

Summary

DOORS® current support for importing documents and linking to documents by inserting an OLE Object did not support our needs for tracking requirements. The reason an extension of DOORS® capabilities was necessary is that many times, requirements, justification and other additional information that needs to be tracked, is not located within DOORS®. Given this limitation, we still needed a way to link from DOORS® objects to parts within an external document. Linking to parts of a document would eliminate the manual searching process we were trying to get away from by implementing DOORS®. One of the main requirements we had for external linking was the capability to link to text within PDF files. We built on OutDoors' capability for linking to objects within Microsoft® Office files and further tailored it to support linking to bookmarks in PDF files.

REFERENCES

Importing PDF Files Into DOORS®

1. Telelogic Resource Center: Importing PDF Files into DOORS <http://www.telelogic.com/download/article/importing_pdfs_doors.pdf>.

Adobe® Acrobat® Bookmarks

2. *Adobe Acrobat 4.0 Guide*, Working With Bookmarks, pp238, ©1999 Adobe Systems Incorporated.

OutDoors

3. Lewis, Ron, "OutDoors Linking System", 18 May 2001 <http://www.telelogic.com/download/userpresentation/us2001/advanced/doc_lewis.pdf>.

Object Linking and Embedding (OLE)

4. "Common Questions: Object Linking and Embedding, Data Exchange" (Article ID: Q122263), ©2001 Microsoft Corporation <http:// support.microsoft.com / support / kb / articles / Q122 / 2 / 63.asp>.

5. North, Ken, "Understanding OLE", *DBMS*, June 1995 <http://www.dbmsmag.com/9506d13.html>.

Adobe® Acrobat® OLE Support

6. *Acrobat Interapplication Communication Overview* <http://partners.adobe.com/asn/developer/acrosdk/docs/iac/IACOverview.pdf>.

7. *Acrobat Interapplication Communication Reference* <http://partners.adobe.com/asn/developer/acrosdk/docs/iac/IACReference.pdf>

DOORS® OLE Support

8. *DXL Reference Manual v5.0*, OLE Objects, pp446-470, ©2001 Telelogic AB.
9. *DOORS API Manual v5.0*, Working With OLE Objects, pp40, ©2001 Telelogic AB.

Intentionally Left Blank

APPENDIX A

Telelogic Americas 2002
User Group Conference
Presentation Slides

Intentionally Left Blank

PDF Object Linking Extension (POLE)

Stephanie Castillo
Sandia National Laboratories
Albuquerque, NM

1

Overview

- Where We Started - Our Challenge
- Existing Importing/Linking Capabilities Not Enough
- OutDoors Linking System
- Modifications and Additions
- The PDF Object Linking Extension
- Advancements

2

It's Easier to Catch Fish With a POLE

3

Where We Started

- Develop system to track and document all requirements and tests
- Thousands of requirements in over 500 pages of paper-based documents
- Required linking and tracking, to ensure customer requirements met

4

DOORS Linking & Importing

- Linking
  - Lets users identify relationships in data
  - Allows for traceability
  - Users can assess impact of potential changes

5

DOORS Linking & Importing

- Importing
  - Import
    - Six supported Formats
    - Object Linking
    - Maintain in DOORS
    - Links - Establishes relationships between objects
  - Insert OLE Object
    - Any windows based document
    - Document Linking
    - Maintain in original format
    - Links- Establish relationships between an object and a document

6

Why Existing Capabilities Were Not Enough

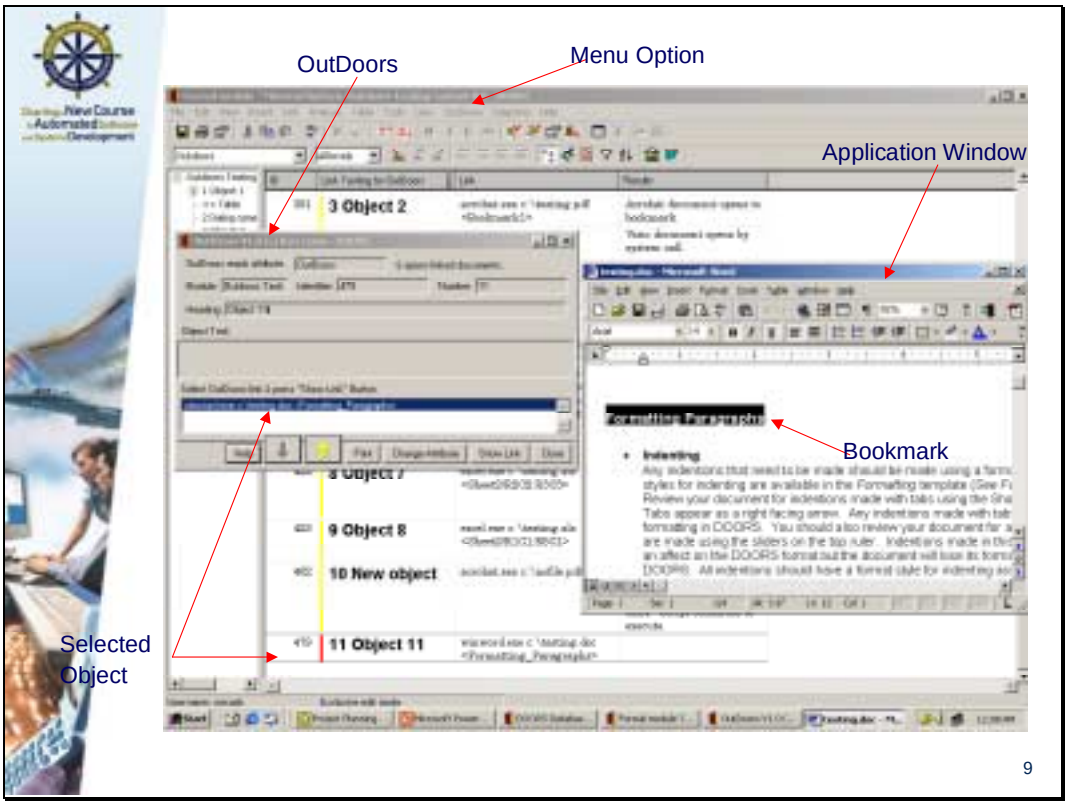
- Not all documents would be imported into DOORS
  - Not One of Six Supported Formats
  - Size (Time Consuming & Costly)
  - Maintenance in Original Format
- Necessity for Relationships to be Created and Maintained to Information ***within*** external documents.
  - Instantaneous Access
  - Easy to Use

7

OutDoors Linking System

- Implemented in DXL
- Run in DOORS environment
- Users easily view external linked data without import into DOORS
  - MSWord:Bookmark
    Excel:Cell
    PowerPoint:Slide
- Documents can be located on a local computer, LAN, intranet or Internet

8



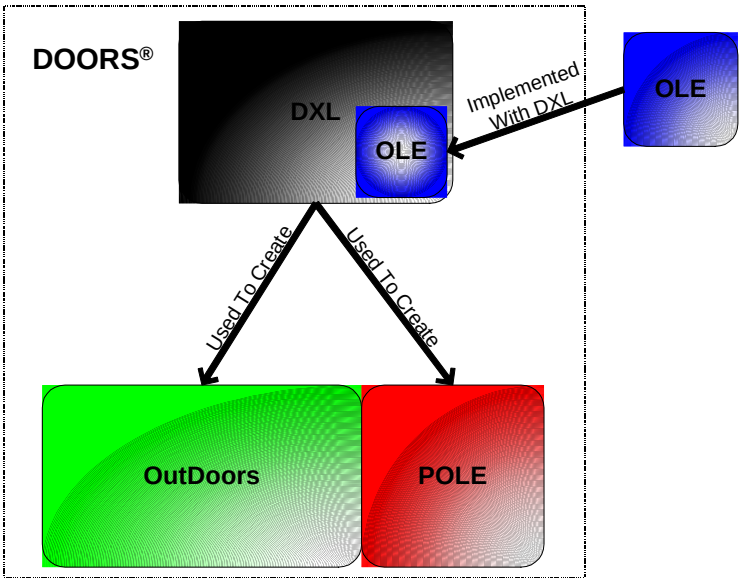
9

Modifications and Additions

- **Why?**
  - **Eliminate Redundancy**
  - **More User Friendly**
  - **Add functionality of linking to bookmarks in PDF**
- **How?**
  - **DXL** - *DOORS eXtension Language*
    - Extend DOORS capability
      - Extend functionality of OutDoors
  - **OLE** - *Object Linking and Embedding*
    - Allows integration of data from different applications
      - Allows Acrobat to respond to a variety of requests from OutDoors

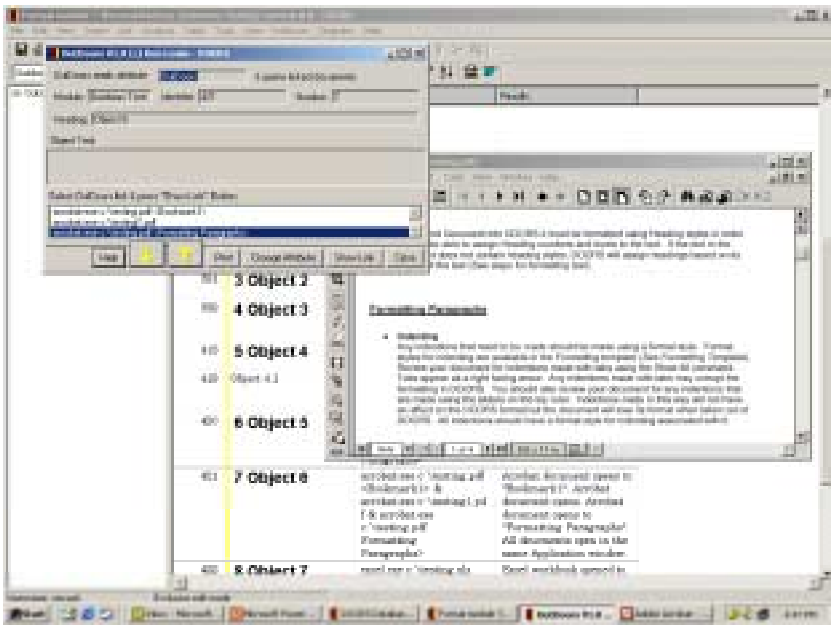
10

How It All Fits Together



11

PDF Object Linking Extension



12

Advancements

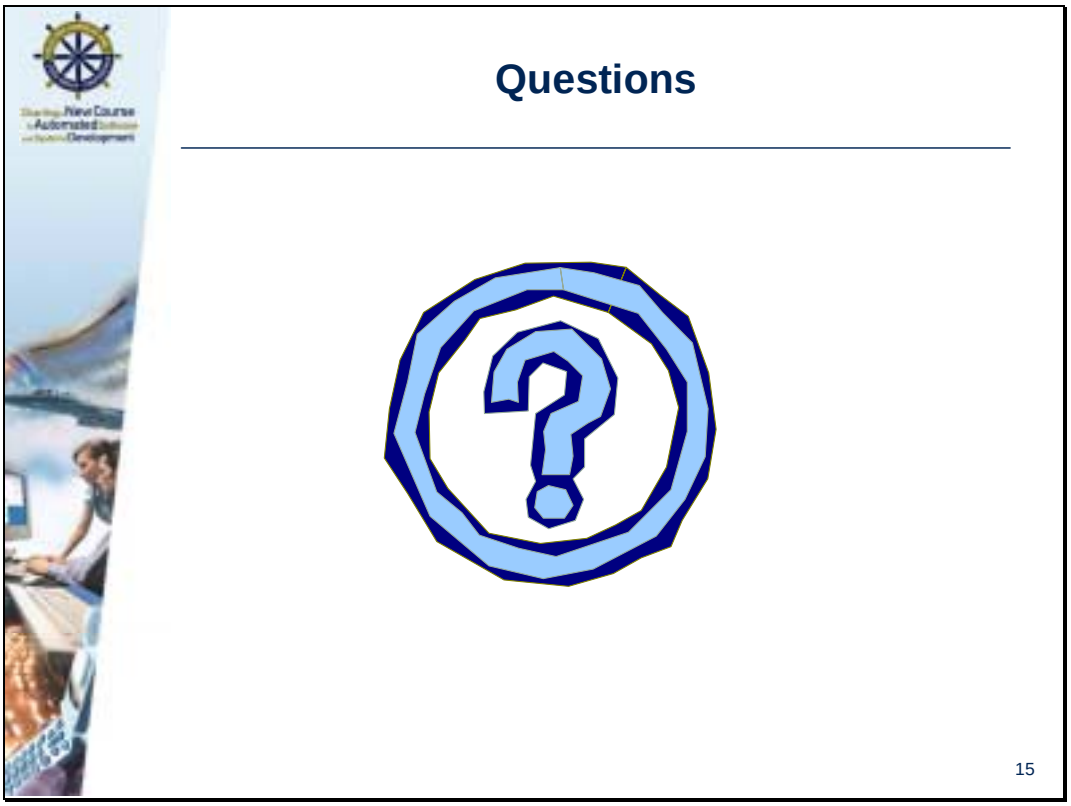
- Link to text in PDF files giving it a differentiating appearance
  - Challenge: Adobe does not currently contain necessary functionality

13

Summary

- PDF Object Linking Extension necessary to maintain traceability between DOORS objects and objects in PDF files.
- POLE was developed by extending onto OutDoors using DXL and OLE.
- Potential Advancements include
  - Differentiating bookmarked text in PDF documents

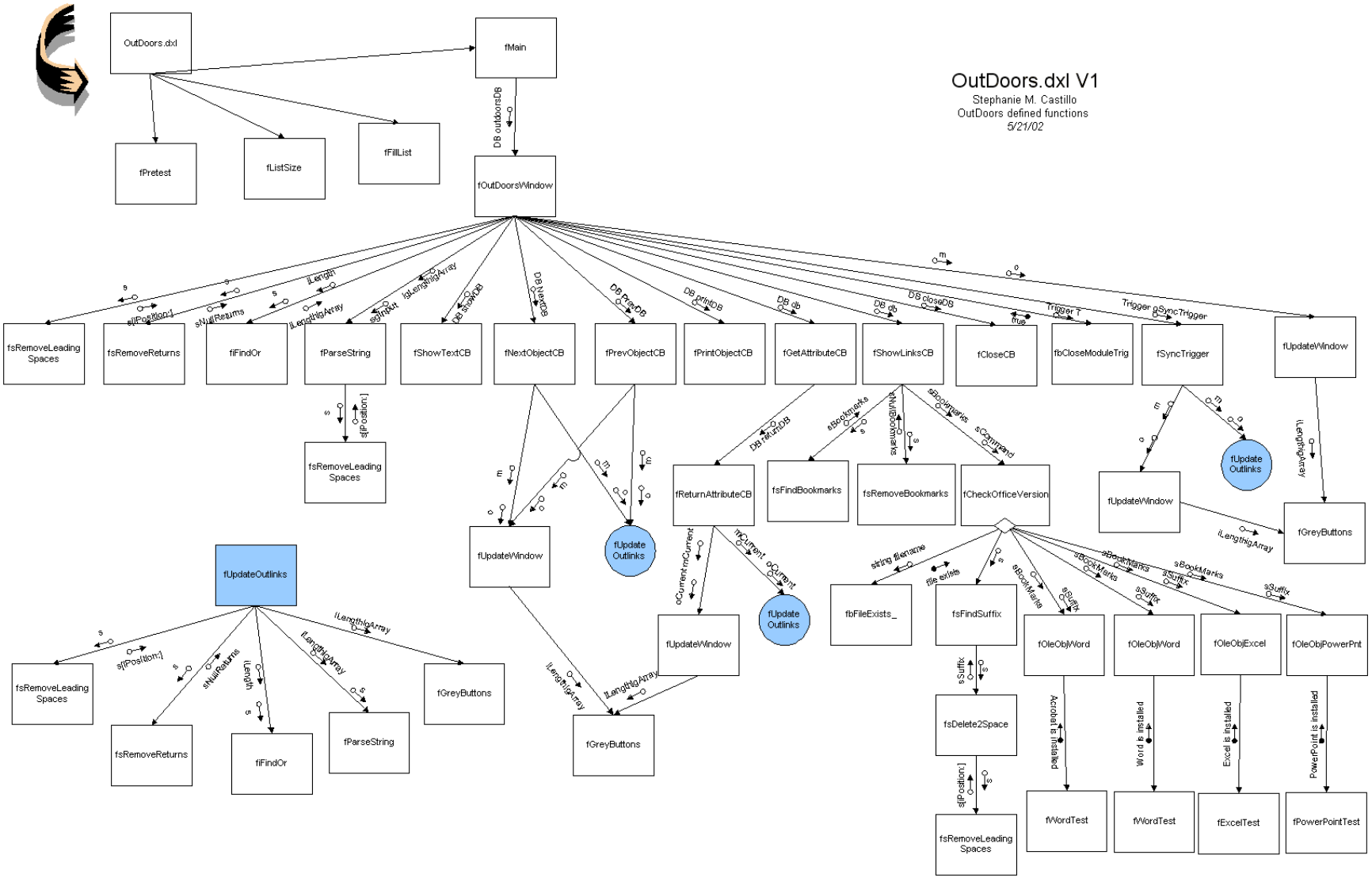
14


Questions
15

APPENDIX B

OutDoors
&
PDF Object Linking Extension
Structured Diagram

Intentionally Left Blank


OutDoors.dxl V1
Stephanie M. Castillo
OutDoors defined functions
5/21/02
OutDoors.dxl
fMain
fPretest
fListSize
fFillList
DB outdoorsDB
fOutDoorsWindow
fsRemoveLeading Spaces
fsRemoveReturns
fiFindOr
fParseString
fShowTextCB
fNextObjectCB
fPrevObjectCB
fPrintObjectCB
fGetAttributeCB
fShowLinksCB
fCloseCB
fbCloseModuleTrig
fSyncTrigger
fUpdateWindow
fUpdate Outlinks
fReturnAttributeCB
fsFindBookmarks
fsRemoveBookmarks
fCheckOfficeVersion
fGreyButtons
fbFileExists_
fsFindSuffix
fsDelete2Space
fOleObjWord
fOleObjWord
fOleObjExcel
fOleObjPowerPnt
fWordTest
fWordTest
fExcelTest
fPowerPointTest
Acrobat is installed
Word is installed
Excel is installed
PowerPoint is installed

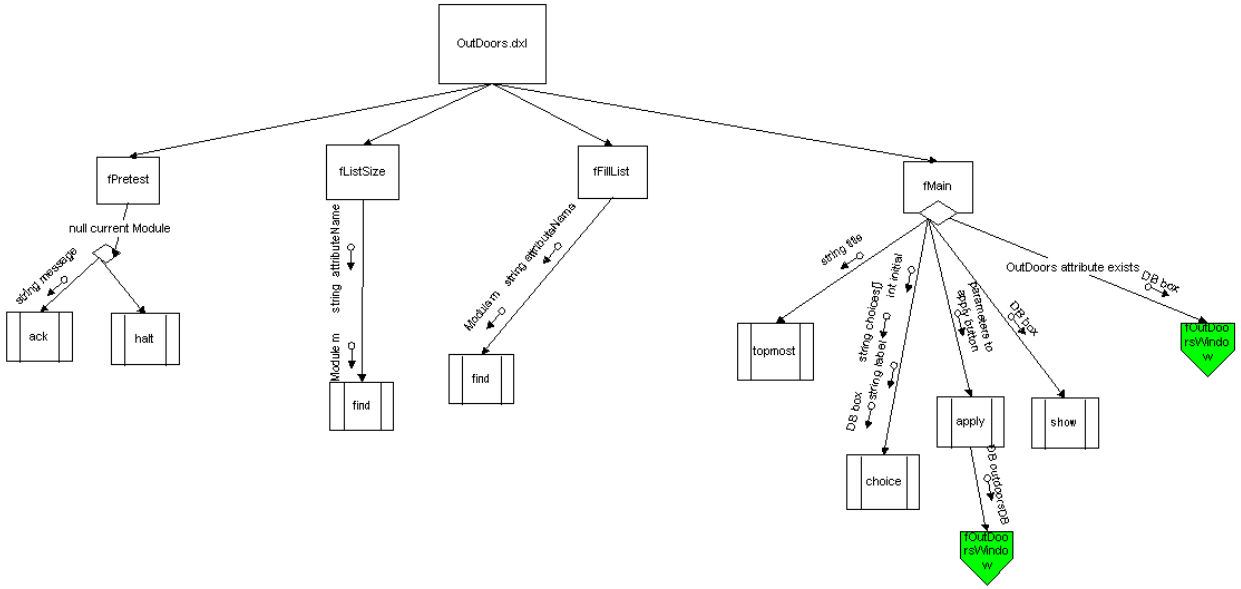
B1

OutDoors Beginning Functions

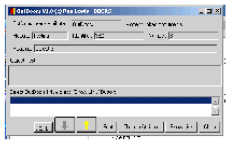
- fPretest
- fListSize
- fFillList
- fMain

OutDoors.dxl V1

Stephanie M. Castillo
OutDoors & DXL Defined Functions
5/21/02

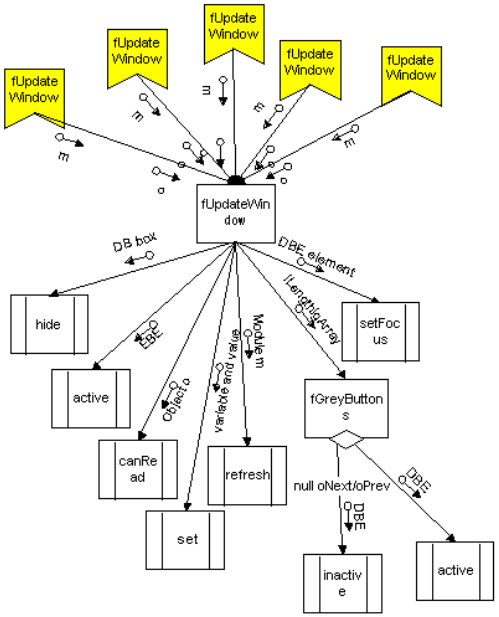


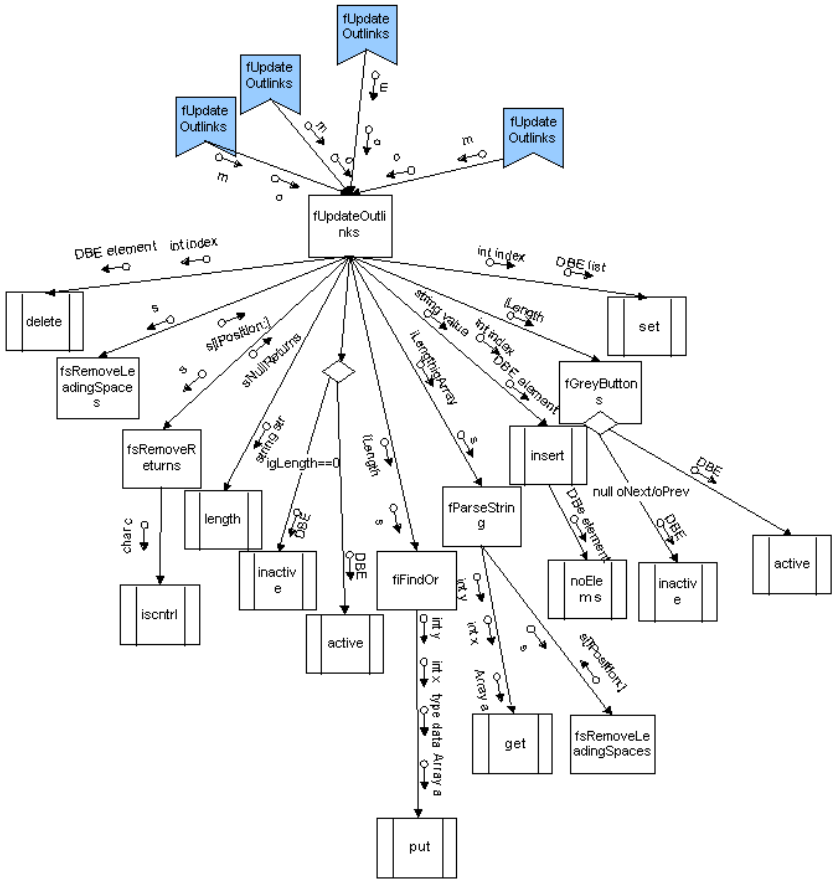
OutDoors Window
- fOutDoorsWindow



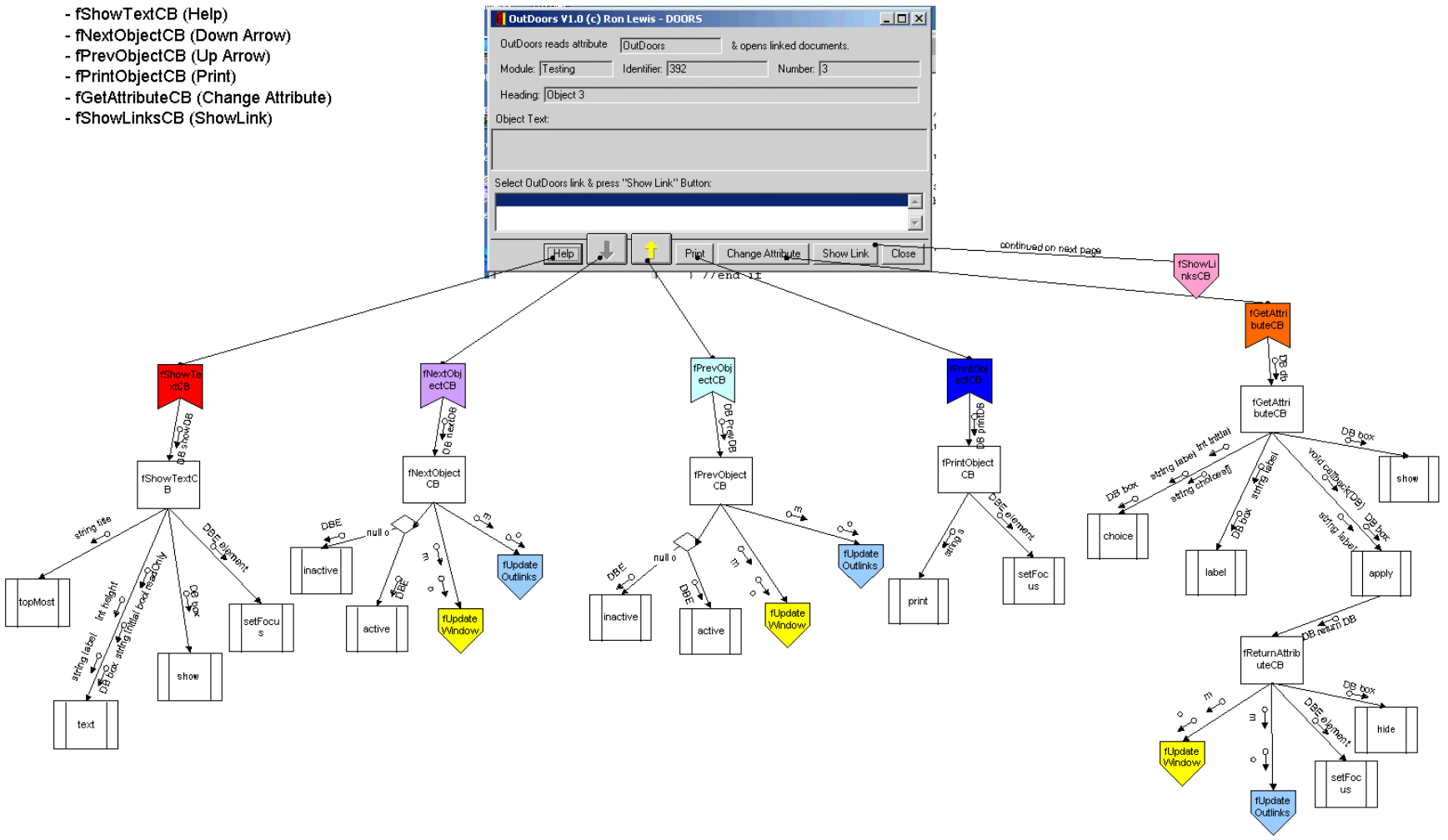
Continued functions

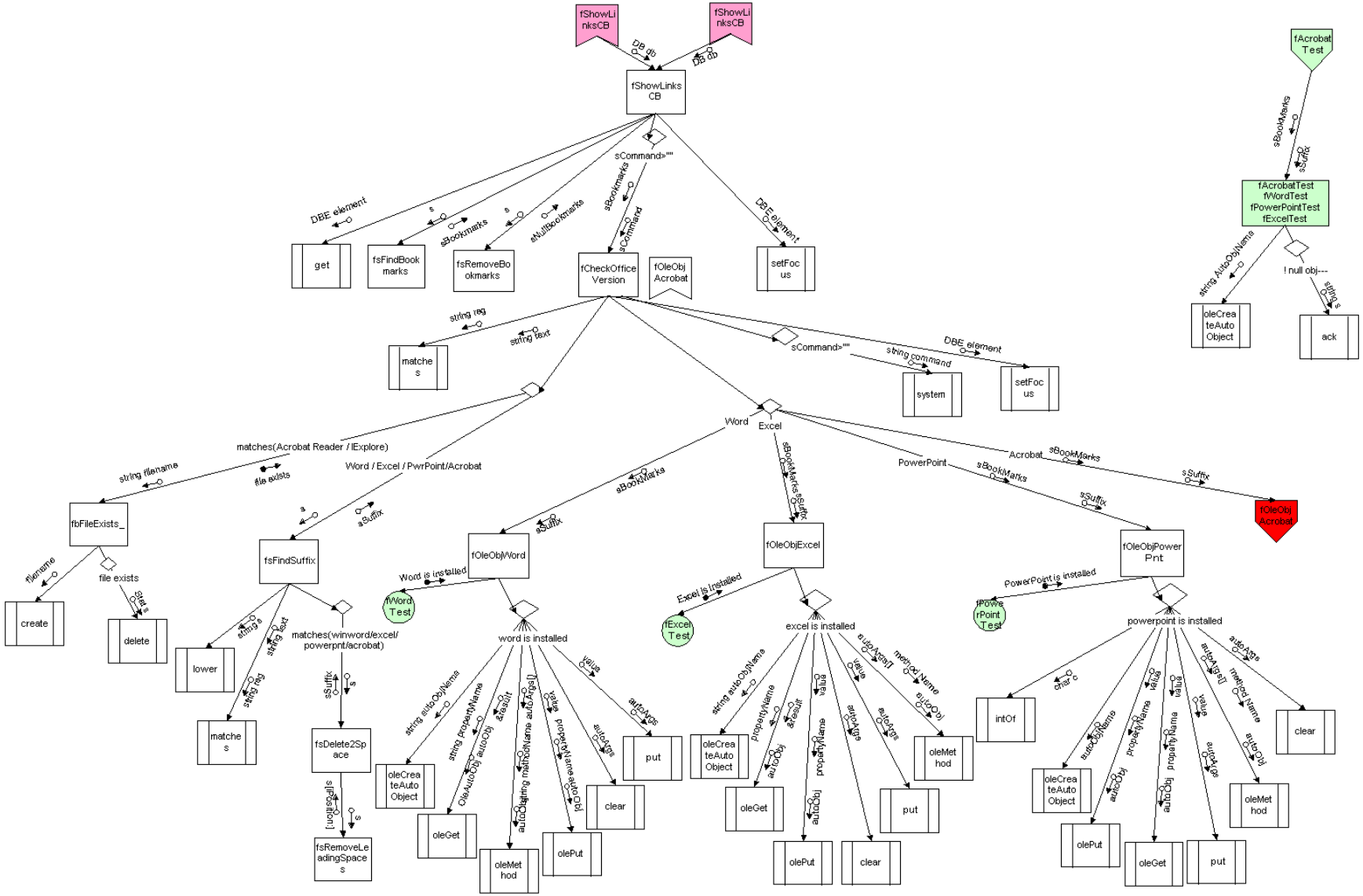
- fUpdateOutlinks
- fUpdateWindow



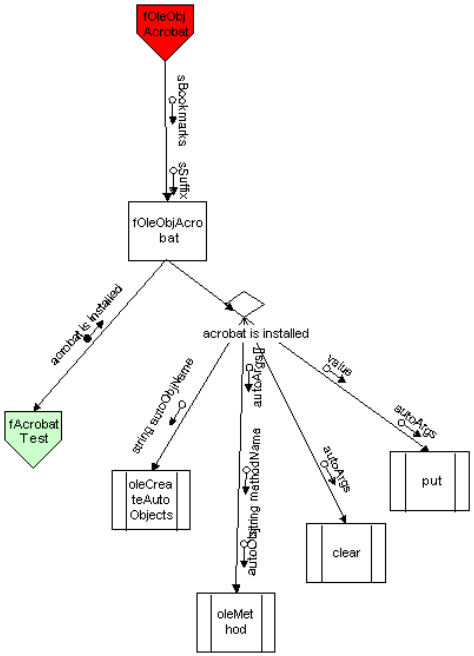


Button Functions
- fShowTextCB (Help)
- fNextObjectCB (Down Arrow)
- fPrevObjectCB (Up Arrow)
- fPrintObjectCB (Print)
- fGetAttributeCB (Change Attribute)
- fShowLinksCB (ShowLink)




fShowLinksCB
fShowLinksCB
DB db
DB db
fShowLinksCB
sCommand=""
DBE element
get
s
sBookmarks
fsFindBookmarks
s
sNullBookmarks
fsRemoveBookmarks
sBookmarks
sCommand
fCheckOfficeVersion
fOleObjAcrobat
DBE element
setFocus
string reg
matches
string text
sCommand=""
string command
system
DBE element
setFocus
matches(Acrobat Reader / IExplore)
string filename
file exists
fbFileExists_
Word / Excel / PwrPoint/Acrobat
a
aSuffix
fsFindSuffix
Word
Excel
PowerPoint
Acrobat
sBookMarks
sSuffix
fOleObjAcrobat
sBookMarks
aSuffix
fOleObjWord
sBookMarks
sSuffix
fOleObjExcel
sBookMarks
sSuffix
fOleObjPowerPnt
filename
file exists
create
delete
string s
string text
string reg
lower
matches
matches(winword/excel/powerpnt/acrobat)
sSuffix
s
fsDelete2Space
string[]
s
fsRemoveLeadingSpaces
Word is installed
fWordTest
word is installed
string autoObjName
string propertyName
aResult
OleAutoObj autoObj
string methodName
autoArgs[]
value
propertyName autoObj
autoArgs
clear
put
autoArgs
oleCreateAutoObject
oleGet
oleMethod
olePut
Excel is installed
fExcelTest
excel is installed
string autoObjName
propertyName
aResult
autoObj
value
propertyName
autoArgs
methodName
autoObj
autoArgs[]
oleCreateAutoObject
oleGet
autoObj
propertyName
autoObj
olePut
autoArgs
oleMethod
put
clear
PowerPoint is installed
fPowerPointTest
powerpoint is installed
char c
intOf
autoObjName
propertyName
value
autoArgs[]
methodName
autoArgs
clear
oleCreateAutoObject
autoObj
propertyName
autoObj
value
autoArgs
oleMethod
olePut
oleGet
put
fAcrobatTest
sBookMarks
sSuffix
fAcrobatTest
fWordTest
fPowerPointTest
fExcelTest
string AutoObjName
! null obj---
string s
oleCreateAutoObject
ack

B6


fOleObj Acrobat
sBookmarks
sSuffix
fOleObjAcro bat
acrobat is installed
acrobat is installed
fAcrobat Test
string autoObjName
autoArgs
oleCrea teAuto Objects
string methodName
autoArgs
oleMet hod
clear
value
autoArgs
put

Distribution:

5	MS 1012	S.M. Castillo, 9511
1	0628	K. de Jong, 2993
1	0501	C.F. Palya, 2305
1	0660	J.A. Larson, 9519
1	1012	G.C. Claycomb, 9511
1	0660	C.D. Harrison, 9515
1	0481	T.F. Hendrickson, 2132
2	0481	B.H. Wohl, 2132
1	9018	Central Technical Files, 8945-1
2	0899	Technical Library, 4916
1	0612	Review & Approval Desk, 9612
		For DOE/OSTI