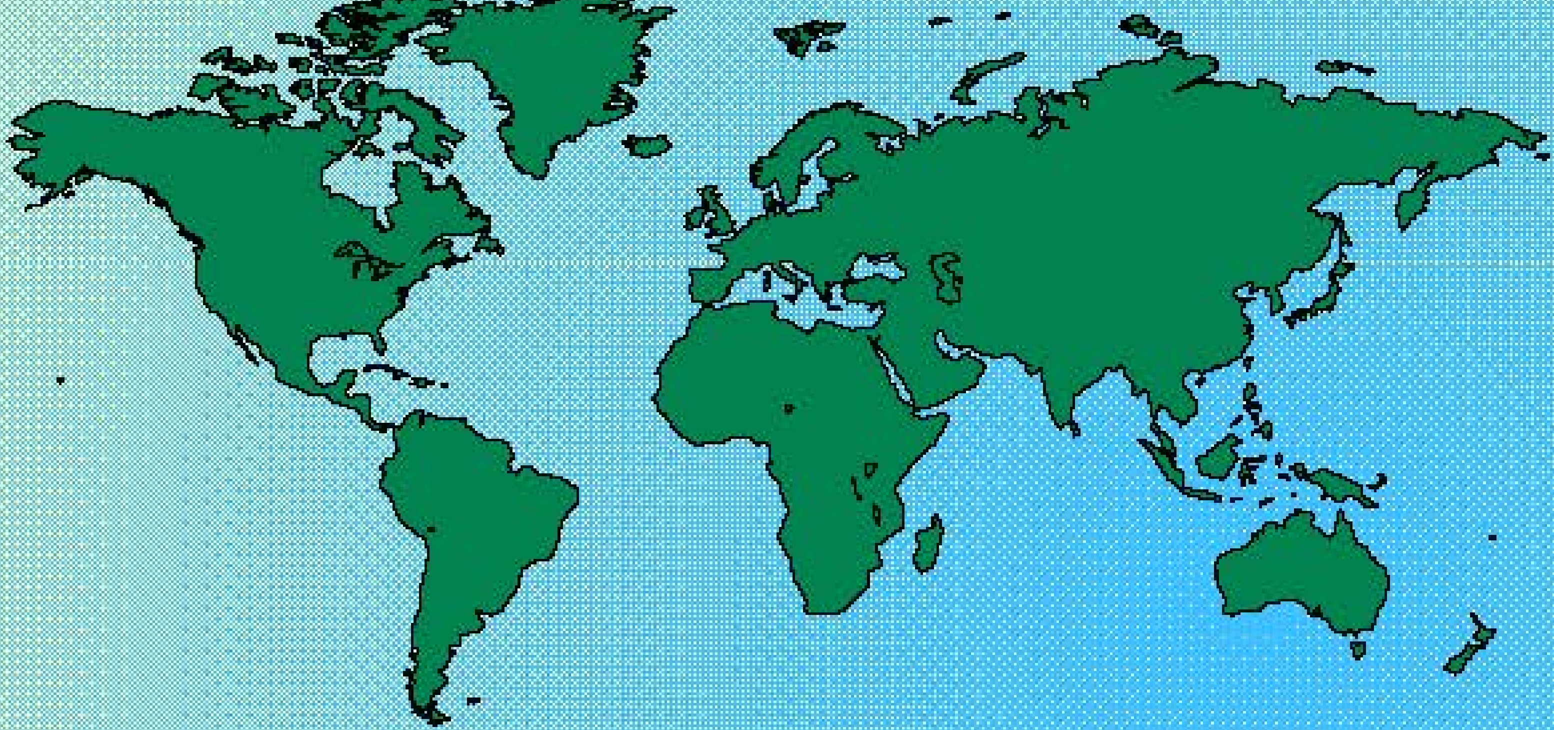
Developing Workplace Shell Applications

(ADOS09)



Sheila A. Harnett, Ph.D.
IBM Boca Raton

Special Notices

- The following terms, used in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:
  - IBM
  - OS/2
  - Win-OS/2
  - C Set++
  - Presentation Manager
  - Workplace Shell
  - System Object Model

Agenda

- What's new with WARP
- WPSH Programming
  - Benefits/Requirements
  - Model
  - Use Items
  - Lifecycle
  - Creating a WPSH class
  - Method categories
  - Customizing behavior
  - Multiple Processes and Threads
- WPSH-related Win and REXX functions
- WPSH DSOM Server
- Example Program

What's new with the OS/2 WARP WPSH

- New Methods and Classes in OS/2 Warp
- New APIs (Win functions and associated Rexx wrappers)
- New setup strings
- SOM 2.X and DSOM
- Updated WPSH Documentation
- Toolkit Availability on Developer's Connection

What can you do with the WPSH API?

- Create a customized desktop configuration
- Integrate your stand-alone application into the data-centric user interaction model
  - data files on desktop can automatically associate to, and launch your .EXE
  - Implement your application as part of WPSH
    - Add new classes to represent the behavioral aspects of your application (templates, new settings, unique context menus, special object and container views, etc)
  - Modify the behavior of existing WPSH classes
    - class replacement

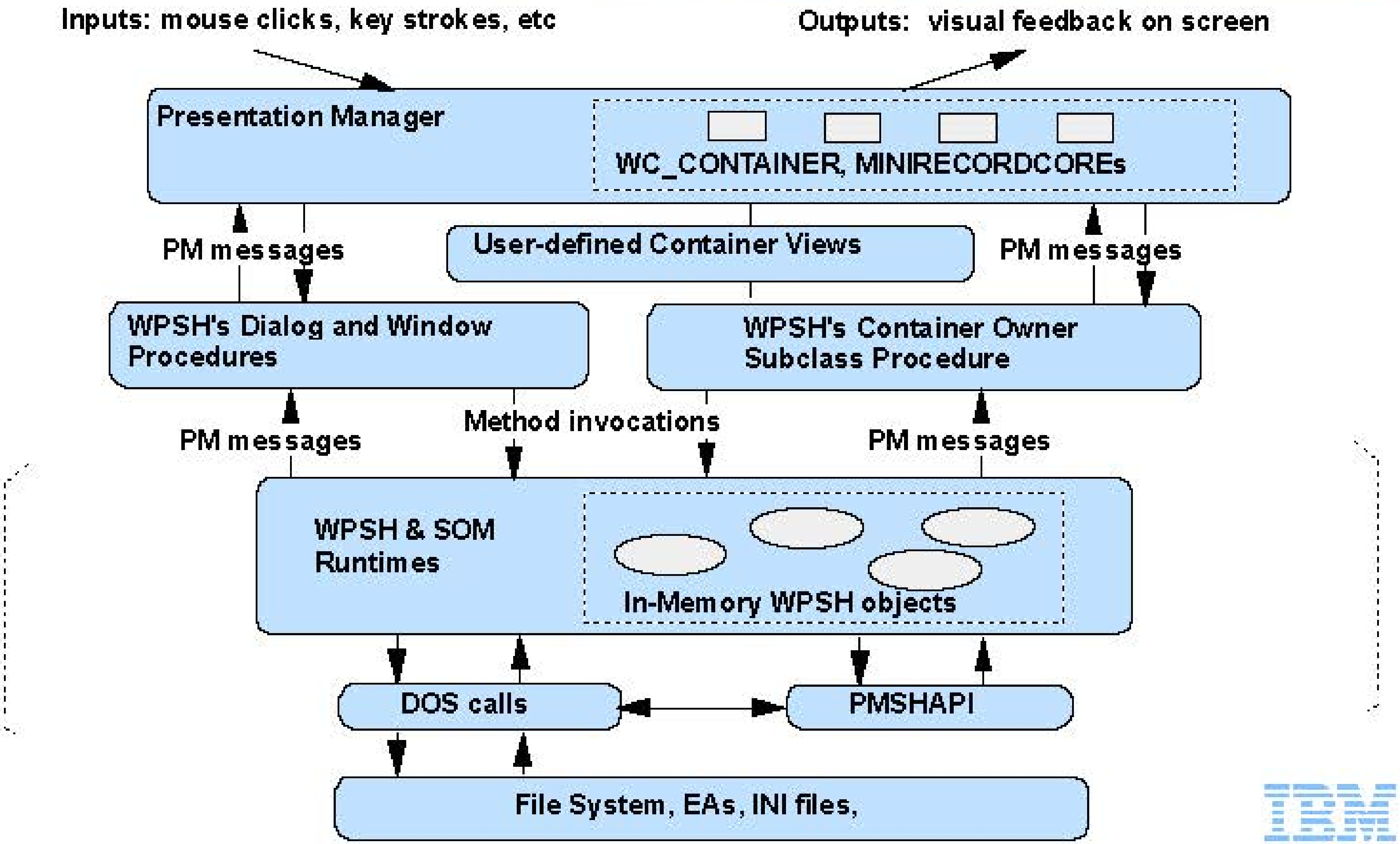
WPSH Programming Pre-requisites

- PM programming concepts
  - Dialogs, controls, containers, window procedures, etc

- Object-Oriented concepts
  - Classes, methods, etc
  - SOM development cycle

- WPSH-specific issues
  - Programming model conventions
  - Method categories and usage

How it all works together...


Inputs: mouse clicks, key strokes, etc
Outputs: visual feedback on screen
Presentation Manager
WC_CONTAINER, MINIRECORDCOREs
User-defined Container Views
PM messages
PM messages
WPSH's Dialog and Window Procedures
WPSH's Container Owner Subclass Procedure
PM messages
Method invocations
PM messages
WPSH & SOM Runtimes
In-Memory WPSH objects
DOS calls
PMSHAPI
File System, EAs, INI files,

OO Review

Class A

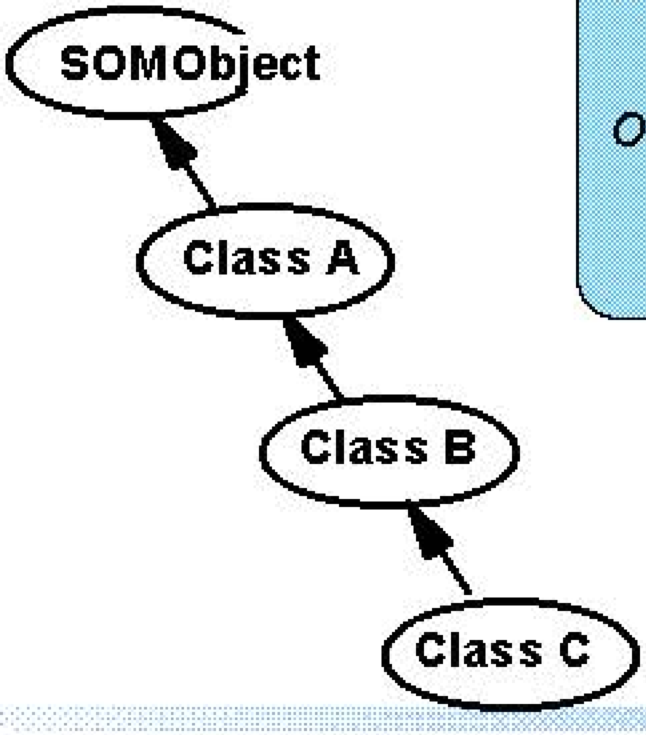
parent: **SOMObject**

data:

iv1, iv2

methods:

m1, m2, m3



Class B

parent: **Class A**

data:

iv3, iv4

methods:

m4, m5

override:

m1

Class C

parent: **Class B**

methods:

m6

override:

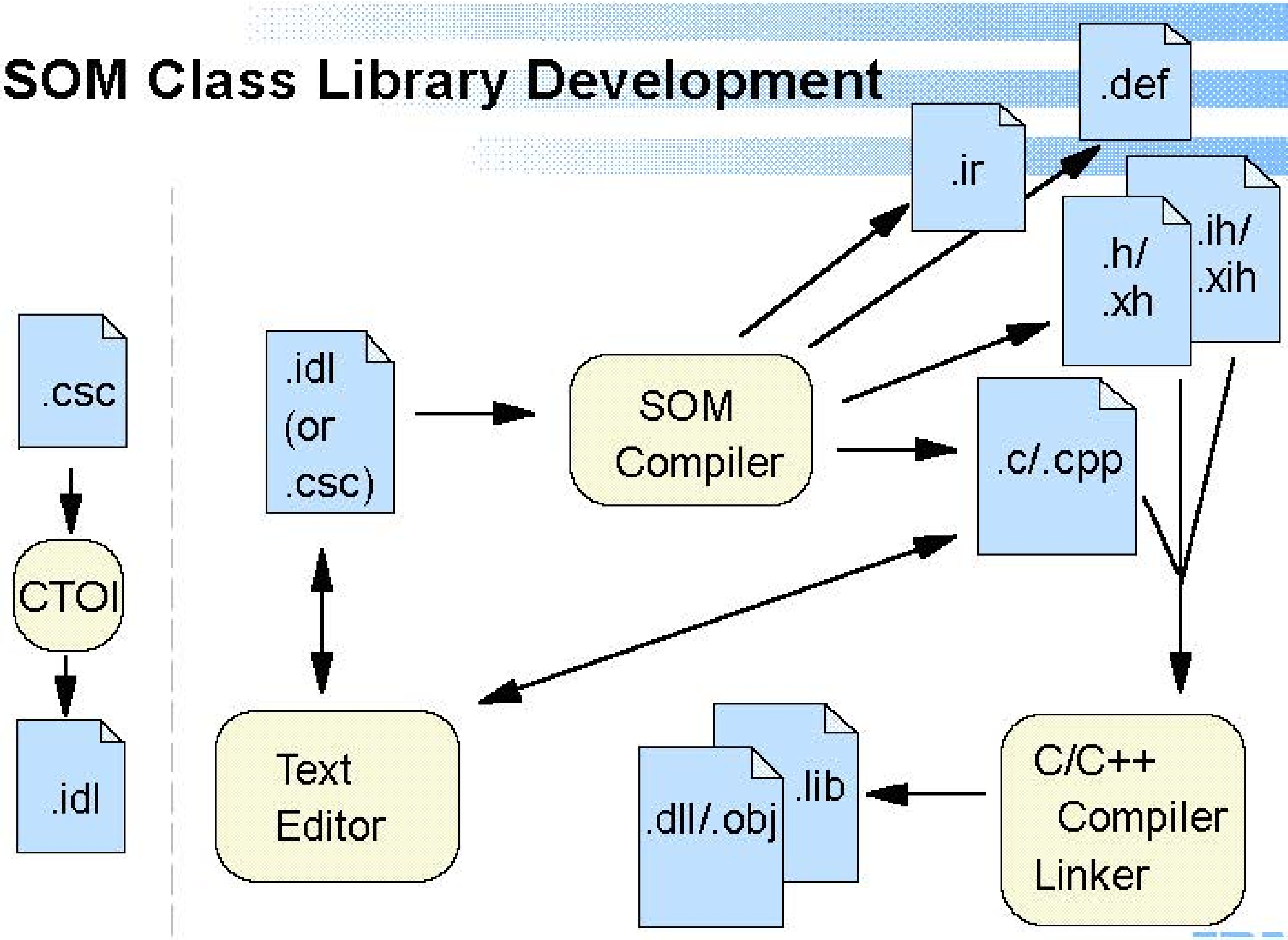
m2, m4

OS/2 WARP WPSH Class Hierarchy

- SOMObject
  - WPObject
    - WPFileSystem
      - WPDataFile
        - WPBitmap
        - WPIcon
        - WPMet *
        - WPProgramFile
    - WPCommandFile
        - WPPif *
        - WPPointer
      - WPFolder
        - WPDesktop
        - WPDrives
        - WPStartup
        - WPTemplates
        - WPNetwork *
        - WPServer *
        - WPNetgrp *
        - WPSharedDir *
        - WPMinWinViewer *
    - WPAbstract
      - WPShredder
      - WPDisk
      - WPProgram
      - WPShadow
        - WPNetLink*
      - WPClock
      - WPCountry
      - WPKeyboard
      - WPMouse
      - WPPalette
        - WPColorPalette
        - WPFontPalette
        - WPSchemePalette
      - WPPower *
      - WPSound
      - WPSystem
      - WPWinConfig *
      - WPLaunchPad *
      - WPPrinter
    - WPTransient
      - WPJob
      - WPPort

* - new in Warp

SOM Class Library Development



.csc

CTOI

.idl

.idl (or .csc)

SOM Compiler

Text Editor

.ir

.def

.h/ .xh

.ih/ .xih

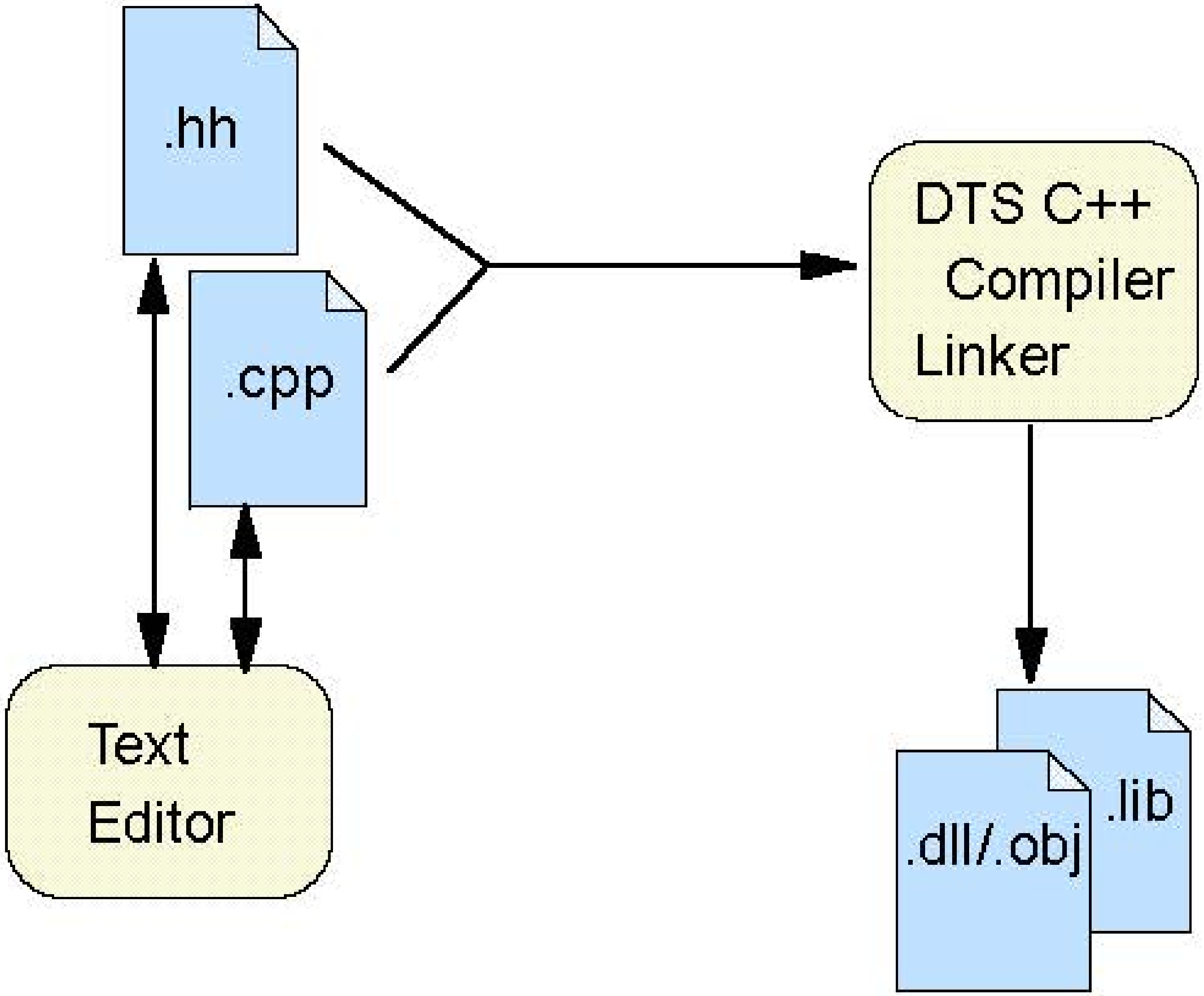
.c/.cpp

C/C++ Compiler Linker

.dll/.obj

.lib

DTS C++ Class Library Development



Converting your .CSC files to .IDL (CTOI)

1) run **CTOI** tool that comes with SOM toolkit
   - create file with all class names defined in your .csc files (e.g.*names.txt*)
   - set CLSFILE env var (e.g. SET CLSFILE=names.txt)
   - copy .sc files from WPSH ancestor classes to current dir

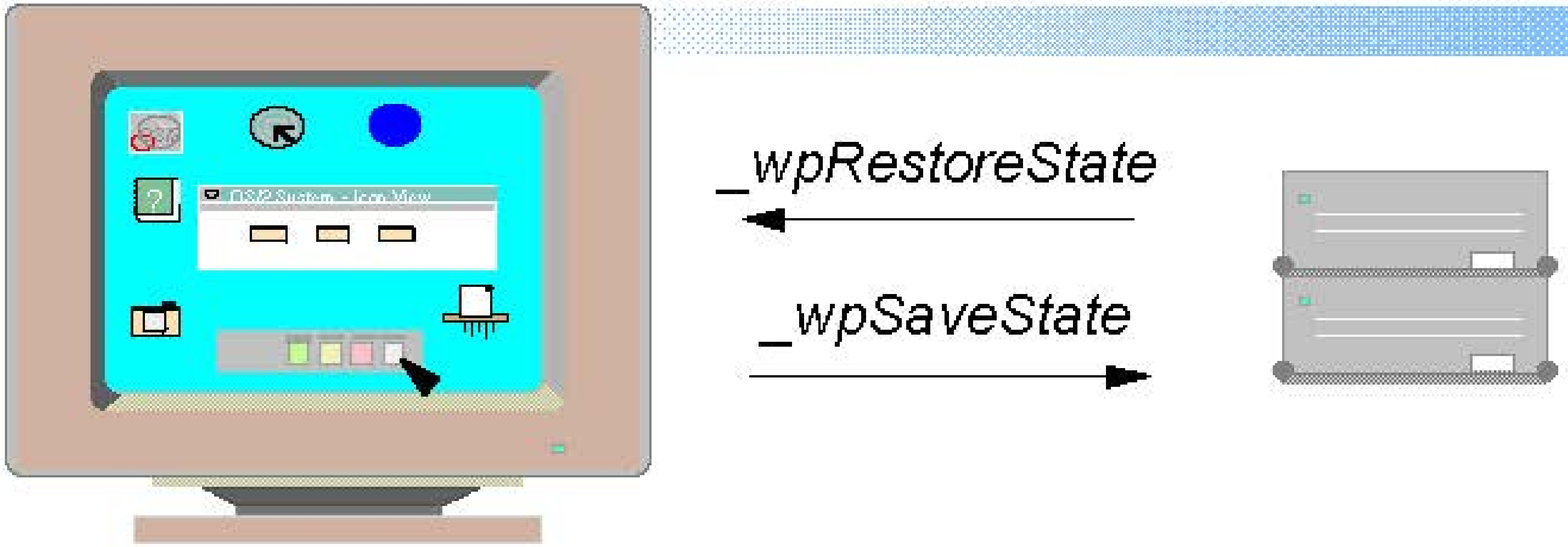
2) fix up your new .idl file, existing .c/.cpp file and makefile
   - .IDL file:
     - = in metaclass' interface definition, *externalprefix* modifier is translated incorrectly (e.g. myf_, but should be myfM_)
     - = change passthru C_ph to C_h with an #ifdef __PRIVATE__
     - = check in/out/inout tags on parameters of your new methods
     - = add type information (optional, see SOM documentation)
   - .C file:
     - = remove #include filename.ph, or replace w/ .h if necessary
   - MAKEFILE
     - = suffixes, SOM pathnames, somtk.lib, -mnoint, -S128000, emit public and private versions of .h/.idl files separately

3) rebuild

Elements of WPSH Programming Model

- Persistence
- Object Life Cycle
- Use Items
- Handles and OBJECTIDs
- Instance Data and Class Data initialization/uninitialization
- Method categories for behavior control
  - settings, settings notebooks, context menus, views, drag/drop behavior, icons, titles, help, etc...

WPObject Persistence

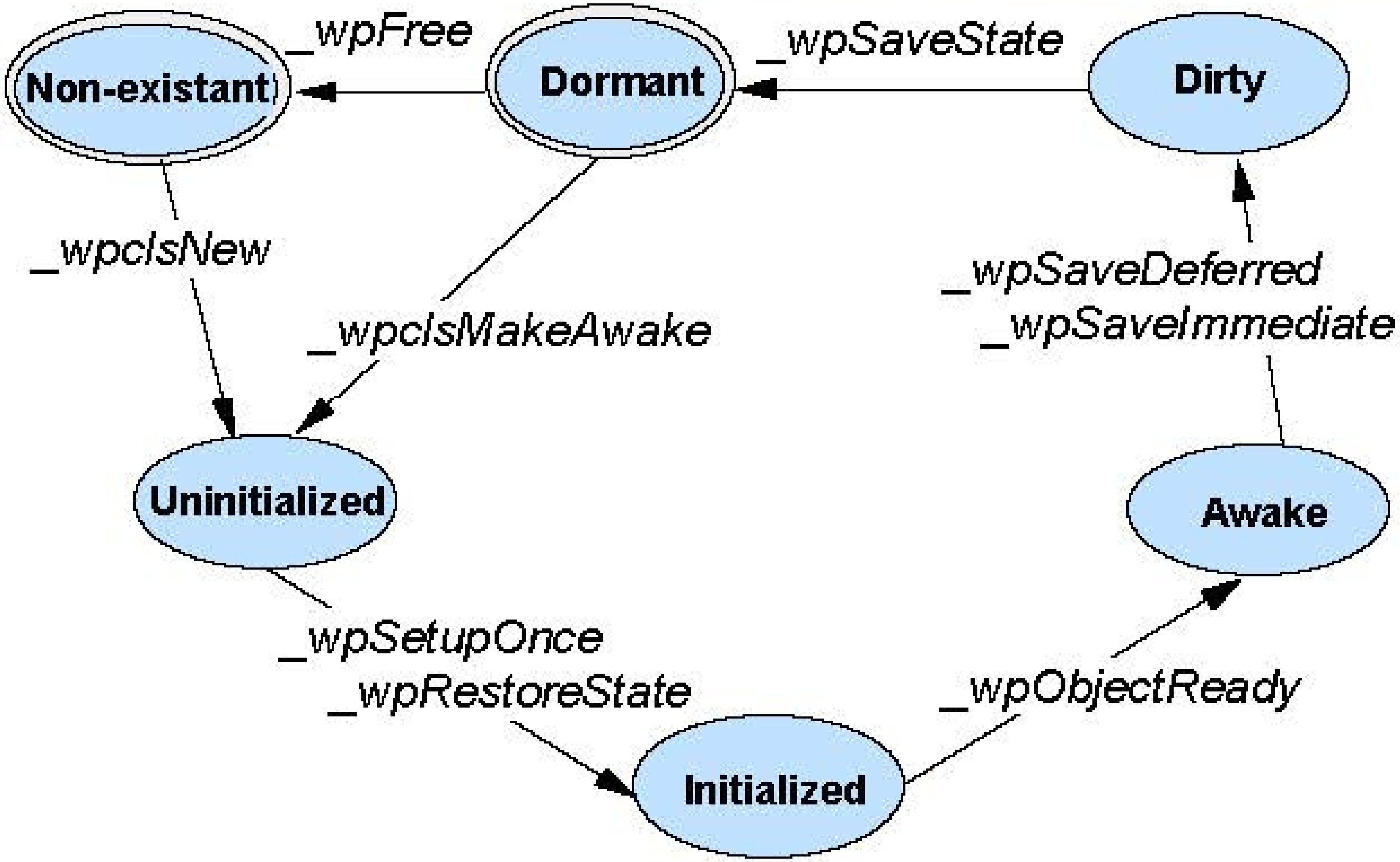


_wpRestoreState

_wpSaveState

- A WPSH object can exist in two states:
  - DORMANT: persistent form of object when it is stored away on disk (WPFileSystem -> EAs, WPAbstract -> INI Files)
  - AWAKENED: object is instantiated as a SOM object that can have methods invoked upon it

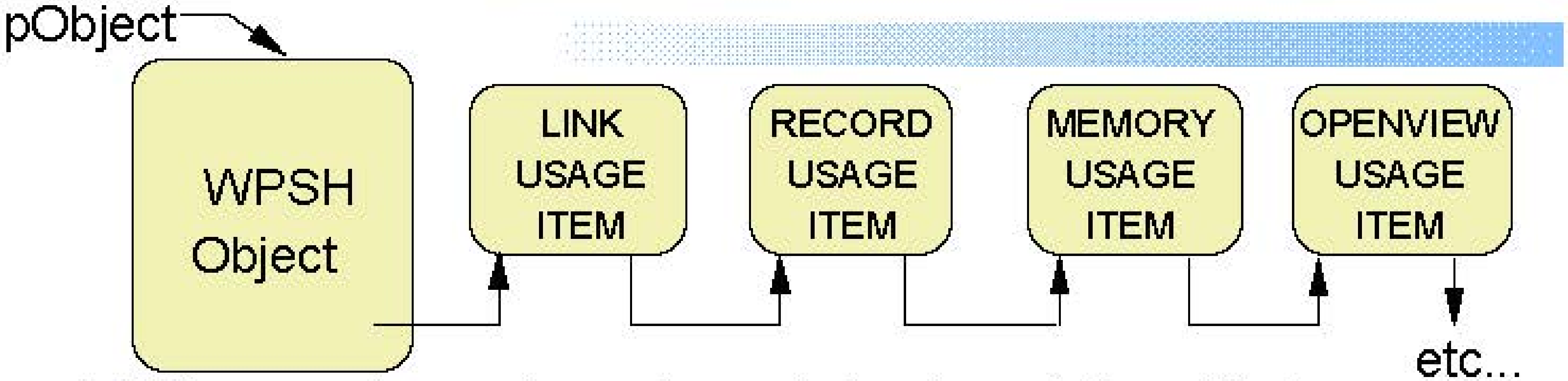
WPObject Life Cycle



WPSH Method Categories

- Object Creation/Initialization
  - wpclsNew
  - wpInitData/UnInitData
  - wpSetupOnce
  - wpSetup
  - wpScanSetupString
  - wpObjectReady
  - wpLockObject/UnlockObject
  - wpclsInitData/UninitData
  - wpclsIncUsage/DecUsage
  - wpDelete
  - wpFree
- Save/Restore of Instance Data
  - wpSaveImmediate
  - wpSaveDeferred
  - wpSave/RestoreState
  - wpSave/RestoreLong
  - wpSave/RestoreString
  - wpSave/RestoreData

Object In-Use List



- LINK - one for each awakened shadow of the object.
- RECORD - one for every container into which the object is inserted. This enables an object to refresh its appearance in all containers at once.
- MEMORY - one for each block of memory allocated via wpAllocMem. Enables WPSH to garbage collect when object is freed.
- OPENVIEW - one for each open view of the object.

Each USEITEM followed by type-specific structure

Method Categories (cont'd)

- Object UseItems
  - wpAddToObjUseList
  - wpDeleteFromObjUseList
  - wpFindUseItem
  - wpFindViewItem

- Memory Management
  - wpAllocMem
  - wpFreeMem

- View Management
  - wpViewObject
  - wpOpen
  - wplose
  - wpRegisterView
  - wpSwitchTo
  - wpclsQueryDefaultView
  - wpCnrInsertObject
  - wpCnrRefreshDetails
  - wpSetConcurrentView
  - wpQueryCoreRecord

Method Categories (cont'd)

- Settings Notebooks
  - wpAddObjectGeneralPage
  - wpAddSettingsPages
  - wpInsertSettingsPage

- Context Menus
  - wpModifyPopupMenu
  - wpInsertPopupMenuItem
  - wpFilterPopupMenu
  - wpMenuItemSelected
  - wpMenuItemHelpSelected
  - wpDisplayMenu

- Folder Management
  - wpAddToContent
  - wpDeleteFromContent
  - wpPopulate
  - wpQueryContent

- Set/Query Object Information
  - wpSet/QueryDefaultHelp
  - wpSet/QueryIconData
  - wpSet/QueryStyle
  - wpSet/QueryTitle
  - wpQueryFolder

Method Categories (cont'd)

- Direct Manipulation
  - wpFormatDragItem
  - wpRender
  - wpRenderComplete
  - wpDragOver
  - wpDrop
  - wpEndConversation

  - wpDraggedOverObject
  - wpDroppedOnObject
- Details Data Managment
  - wpclsQueryDetailsInfo
  - wpQueryDetailsData
  - wpCnrRefreshDetails

- Base Class Find Support
  - wpclsFindObjectFirst
  - wpclsFindObjectNext
  - wpclsFindObjectEnd

Method Categories (cont'd)

- Some Class Methods
  - wpclsQueryStyle
  - wpclsQueryIconData
  - wpclsQueryIconDataN
  - wpclsQueryDefaultHelp
  - wpclsQueryTitle

Settings Notebook Customization

- Creating and adding your own settings pages:
- for each new page, write a DlgProc and create a **_AddXXXXXPage** method that calls **_wpInsertSettingsPage** with its own PAGEINFO
- override **_wpAddSettingsPages**:

```
if (parent_wpAddSettingsPages(somSelf, hwndNotebook)
          && _AddMyNewPage1( somSelf, hwndNotebook)
          && _AddMyNewPage2(somSelf, hwndNotebook))
{
      return TRUE ;
}
else
{        /* Insert failed */
       return FALSE;
}
```

Settings Notebook Customization (cont'd)

- override **_wpclsQuerySettingsPageSize** if your pages are larger than the default notebook size

- To remove any existing pages:
  - override chosen settings page method (**_wpAddObjectGeneralPage, _wpAddObjectWindowPage, _wpAddFile1Page, _wpAddFile2Page, _wpAddFile3Page**, etc...)
  - and simply return

    SETTINGS_PAGE_REMOVED
  - don't call parent

Context Menu Customization

- To remove an item, override **_wpFilterPopupMenu**:

  return(parent_wpFilterPopupMenu(......) &
  ~CTXT_xxxx)

  - NOTE: if what you really mean is to change the object's style, override **_wpclsQueryStyle**

- To add your own items, override **_wpModifyPopupMenu**:

  call **_wpInsertPopupMenuItems** for each new item, and then call parent

- For new menu items, override **_wpMenuItemSelected** and **_wpMenuItemHelpSelected**

Saving/Restoring your instance data

- override **_wpSaveState** to save instance data that you want to persist
  - define unique IDKEYs for each piece of instance data and call appropriate **_wpSaveLong/String/Data** method

- override **_wpRestoreState** to restore your data
  - call appropriate **_wpRestoreLong/String/Data** method with each IDKEY for each stored value

- call parent in both cases

Customizing Setup String Processing

- To add your own "remote-control ability," override the **_wpSetup** method:
  - Create your own ***key=value*** pairs, using **_wpScanSetupString** to parse
  - call parent
- **_wpSetup** is triggered by **WinSetObjectData** and **SysSetObjectData** APIs
- **_wpSetupOnce** is called by **_wpclsNew**, when an object is first instantiated, (**_wpSetupOnce** calls **_wpSetup**)
- Note the difference between **_wpSetupOnce** vs. _w**pSetup)**

Adding your own Class Details

- override **_wpclsQueryDetailsInfo**
  - returns chain of CLASSFIELDINFO structures, representing the "meta" information about the class' details (e.g. type, length and offset of each details field)
  - call parent, and add your CLASSFIELDINFO at end of chain
  - return sum of your and your parent's columns
  - override **_wpclsInitData** to initialize your CLASSFIELDINFO once
- override **_wpQueryDetailsData**
  - returns linked list of actual details data for object
- call **_wpclsSetDetailsClass**
  - allows a folder to display the details columns for with details from a user-defined class
- call **_wpCnrRefreshDetails**
  - call when updating a piece of details data, it will refresh all open views

Customizing Print Behavior

- Override the **_wpPrintObject** method
  - invoked from context menu, and when object dropped on Printer
- **wpPrintObject** starts a new thread to handle printing, the thread chooses an appropriate print method based on the type of the file
  - **_wpPrintPlainTextFile**
  - **_wpPrintMetaFile**
  - **_wpPrintPifFile**
  - **_wpPrintPrinterSpecificFile**
- Make sure objects of your class have their TYPE set correctly (don't forget your templates)

Drag/Drop Customization

- use **_wpFormatDragItem** to specify your own drag item information (Default is <DRM_OBJECT,DRF_OBJECT>)

```
typedef struct _DRAGITEM          /* drag item */
{
  HWND    hwndItem;               /* conversation partner          */
  ULONG   ulItemID;               /* identifies item being dragged */
  HSTR    hstrType;               /* type of item                  */
  HSTR    hstrRMF;                /* rendering mechanism and format*/
  HSTR    hstrContainerName;      /* name of source container      */
  HSTR    hstrSourceName;         /* name of item at source        */
  HSTR    hstrTargetName;         /* suggested name of item at dest   */
  SHORT   cxOffset;               /* x offset of the origin of the */
  /*                                 image from the mouse hotspot   */
  SHORT   cyOffset;               /* y offset of the origin of the */
  /*                                 image from the mouse hotspot   */
  USHORT  fsControl;              /* source item control flags     */
  USHORT  fsSupportedOps;         /* ops supported by source       */
} DRAGITEM;
typedef DRAGITEM *PDRAGITEM;
```

Drag/Drop Customization (cont'd)

- override **_wpDrop** to perform special processing on items being dropped
- override **_wpDragOver** to let target accept or refuse drop

```
typedef struct _DRAGINFO     /* draginfo */
{
  ULONG    cbDraginfo;       /* Size of DRAGINFO and DRAGITEMs  */
  USHORT   cbDragitem;       /* size of DRAGITEM                */
  USHORT   usOperation;      /* current drag operation          */
  HWND     hwndSource;       /* window handle of source         */
  SHORT    xDrop;            /* x coordinate of drop position   */
  SHORT    yDrop;            /* y coordinate of drop position   */
  USHORT   cditem;           /* count of DRAGITEMs          */
  USHORT   usReserved;       /* reserved for future use         */
} DRAGINFO;
typedef DRAGINFO *PDRAGINFO;
```

Sequence of Direct Manipulation events*

- Sequence of interactions:

Method:	Invoked On:
_wpFormatDragItem	source
_wpDragOver	target
_wpDrop	target

If source rendering is necessary, target initiates:

_wpRender	source
_wpRenderComplete	target
_wpEndConversation	source

Customizing Associations and Types

- Create a WPDataFile subclass for files on which your application operates:
  - Define a file type that your application understands (e.g. Spreadsheet)
    - override **_wpclsQueryInstanceType**
  - Define a file name filter that your application deals with (e.g. *.SPD) that the application deals with
    - override **_wpclsQueryInstanceFilter**
  - Give your application's .EXE an .ASSOCTABLE EA with entries for your file types and filters that you want to pick up your association

Managing Your Class's Templates

- To control the creation of templates for your class:
  - override **_wpclsCreateDefaultTemplates**
    - create your templates in your own folder
    - give your templates OBJECTIDs so your de-install program can remove them easily
    - You may want to use CLSSTYLE_NEVERTEMPLATE during development
  - Make sure you set your .TYPE EA on your templates

Creating Your own Views

- If you add your own views, make your view menu item id is the same as your view's id, so the checkmark on the dropdown open submenu will be correct
- Override **_wpOpen**, recognize your special OPEN_xxxx flag
- Make sure you call **_wpRegisterView**

WPSH and the Container Control

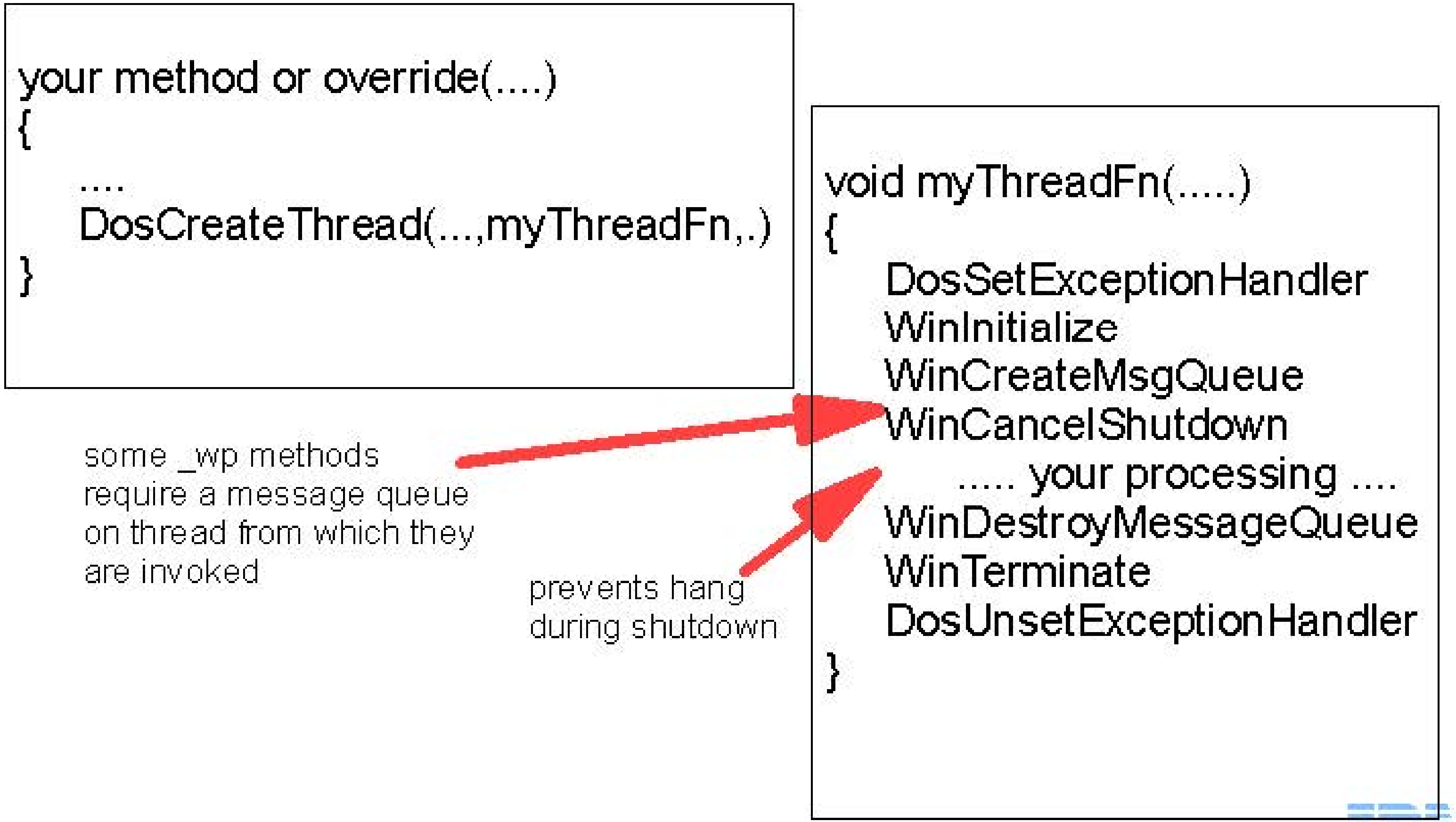
- CUA container control is the primary way of interacting with Workplace Shell objects
- All objects with which the user interacts are simply records that have been inserted into a container control
- Any workplace object can be inserted into any container control created on the Workplace process using the **_wpCnrInsertObject** method
- Only one MINIRECORDCORE per object is used, even if the object exists in multiple views (access by calling **_wpQueryCoreRecord**)
- Since we subclass the container control's owner, you can only have ONE container control into which you are inserting objects per owner. This means you may have to expand your window hierarchy.

WPSH Container Owner Subclass Proc

- When an object is inserted into a container via **_wpCnrInsertObject**, we subclass the owner of the container
  - provides support that maps container records to their WPObject counterparts, and related behaviors (context menus, drag/drop support)
- When a view is registered, we subclass the frame
  - adds the viewed object's context menu to the system menu of the view
  - assumes that FID_CLIENT is a container and that the container's owner is a frame window
  - You may need to extend window hierarchy to make work correctly

Use Multi-threading where possible

- Off-load I/O intensive operations to a separate thread



```
your method or override(....)
{
    ....
    DosCreateThread(...,myThreadFn,.)
}
```

```
void myThreadFn(.....)
{
    DosSetExceptionHandler
    WinInitialize
    WinCreateMsgQueue
    WinCancelShutdown
        ..... your processing ....
    WinDestroyMessageQueue
    WinTerminate
    DosUnsetExceptionHandler
}
```

some _wp methods require a message queue on thread from which they are invoked

prevents hang during shutdown

WPSH Processes and Threads

- Shell Process
  - Application Starter Thread
  - Shutdown Thread
- Workplace Process
  - Object Thread
  - Lazy Writer Thread
  - Sleepy Time Thread
  - File System Notification Threads
  - Threads that come and go:
    - Populate, Asynchronous refresh, Closing Folders, Handling Copy/Move tasks, Printing, Finding
  - WPSH DSOM Server Thread

WPSH-related RexxUtil APIs

- SysCopyObject
- SysCreateShadow
- SysMoveObject
- SysCreateObject
- SysDestroyObject
- SysSetObjectData
- SysOpenObject
- SysSaveObject
- SysRegisterObjectClass
- SysDeregisterObjectClass

To use, always place following lines at top of Rexx file:

```
/* */
call RxFuncAdd 'SysLoadFuncs', 'RexxUtil', 'SysLoadFuncs'
call SysLoadFuncs
```

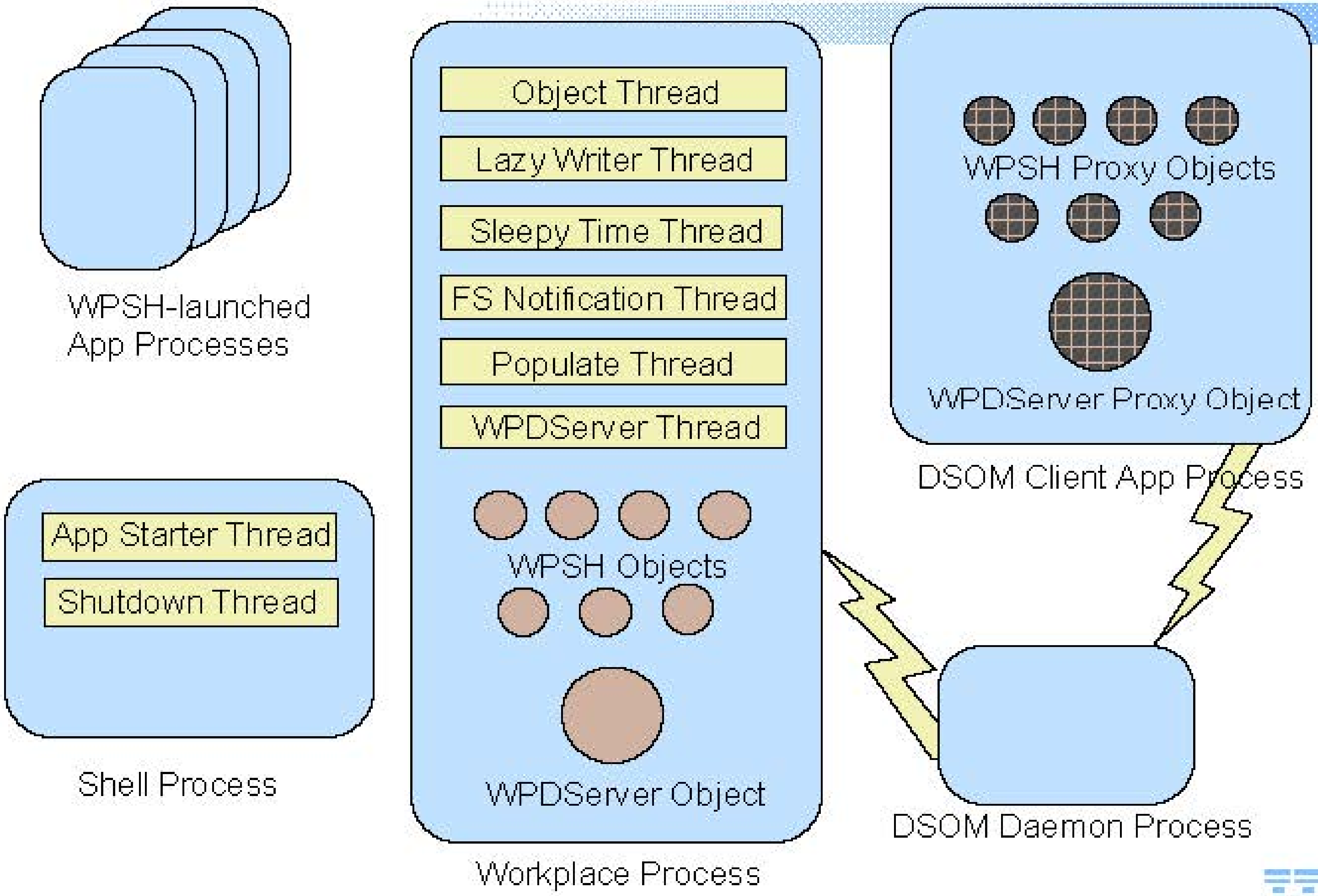
WPSH-related Win APIs

- WinCopyObject
- WinCreateShadow
- WinCreateShadow
- WinMoveObject
- WinOpenObject
- WinSaveObject
- WinQueryActiveDesktopPath
- WInQueryObjectPath
- WinCreateObject
- WinQueryObject
- WinDestroyObject
- WinSetObjectData
- WinRegisterObjectClass
- WinDeregisterObjectClass
- WinReplaceObjectClass
- WinEnumObjectClasses
- WinStoreWindowPos
- WinRestoreWindowPos
- WinLoadFileIcon
- WinSetFileIcon
- WinFreeFileIcon
- WinShutdownSystem

WPSH DSOM Server-related Win APIs

- *WinIsSOMDDReady*
- *WinIsWPDServerReady*
- *WinRestartSOMDD*
- *WinRestartWPDServer*

WPSH DSOM Server Access



Example WPSH DSOM Client App

```
#define INCL_DOS
#define INCL_WINWORKPLACE
#define INCL_PM
#include <os2.h>
#include <somd.h> /* DSOM headers */
#include <somtcnst.h>

/* include header for each WPSH class you will be working with */
#include <wpobject.h>
#include <wpdesk.h>
#include <wpclsmgr.h>                          /* WPSH Class Mgr */
WPClassManager *vWPClassManagerObject;

enum { STOP, START };

int main( int argc, char *argv[] )
{
  Environment     *ev;
  SOMClass        *Server;
  SOMObject       *pDesktopObj;
  M_WPFolder      *pWPFolderClass;
  BOOL             bDaemonUp;
  BOOL             bServerUp;
```

Example WPSH DSOM Client (cont'd)

```
  bDaemonUp = WinIsSOMDDReady();
  if( !bDaemonUp ) WinRestartSOMDD( START );

  bServerUp = WinIsWPDServerReady();
  if( !bServerUp ) WinRestartWPDServer(START);

  ev = SOM_CreateLocalEnvironment();
  SOMD_Init( ev );
  vWPClassManagerObject = WPClassManagerNew();
  _somMergeInto( SOMClassMgrObject, vWPClassManagerObject );

  /* Initialize all the Workplace Shell Classes you will be working with */
  WPFolderNewClass( 1, 1 );

  Server = _somdFindServerByName( SOMD_ObjectMgr, ev, "wpdServer" );

  pWPFolderClass = _somdGetClassObj( Server, ev, "WPFolder");

  pDesktopObj = _wpclsQueryFolder( pWPFolderClass, "<WP_DESKTOP>", TRUE );

/* invoke other methods.... */
```

Example WPSH DSOM Client (cont'd)*

```
    _somdReleaseObject( SOMD_ObjectMgr, ev, Server );

    SOMD_Uninit( ev );
    SOM_DestroyLocalEnvironment( ev );
  }
```

Sample Application

- MYFILE and MYFOLDER classes
- REXX scripts for install/deinstall
- Documented source files, ipf files, rc files, etc
- SOM 2.X IDL files
- MYFILE:
  - adds class details, unique open view, only droppable on MYFOLDER, adds settings page, modifies context menu
- MYFOLDER:
  - manipulates Use Items, manages view handles, performs cleanup prior to first opening, defaults to details view, allows concurrent views, accepts drops of MYFILE objects, defines custom open icon,...

References

- ***OS/2 Warp WPSH Reference and Programming Guide***
- ***OS/2 Warp Workplace Shell API;*** Pollack
- *OS/2 Developer Magazine*
- ***OS/2 Warp Unleashed;*** Moskowitz and Kerr, et. al.
- *Developer's Connection Newsletter*
- Primer for the OS/2 Workplace Shell Programmer, IBM Technical Report TR 29.1829; Kehn
- Track-A-Snack: A Workplace Shell Programming Example, IBM Technical Report TR 29.1820; Stanger, Melahn and Oakley
- ***Client/Server Programming with OS/2 2.1***, 3rd Ed., Orfali and Harkey
- OS2DEV forums on Compuserve

App A: Some new WPSH methods

- WPObject
  - wpCnrDeleteUseItem
  - wpIsDeleteable
  - wpQueryCoreRecord
  - wpSetObjectID
  - wpQueryObjectID
  - wpCnrRefreshDetails
  - wpFindViewItem
  - wpLockObject
  - wpIsLocked
  - wpObjectReady
  - wpIsObjectInitialized
- WPObject (cont'd)
  - wpCreateAnother
  - wpSaveDeferred
  - wpQueryHandle
  - wpDisplayMenu
  - wpSetupOnce
  - wpclsFindOneObject
  - wpQueryFolder
  - wpclsDecUsage
  - wpclsIncUsage

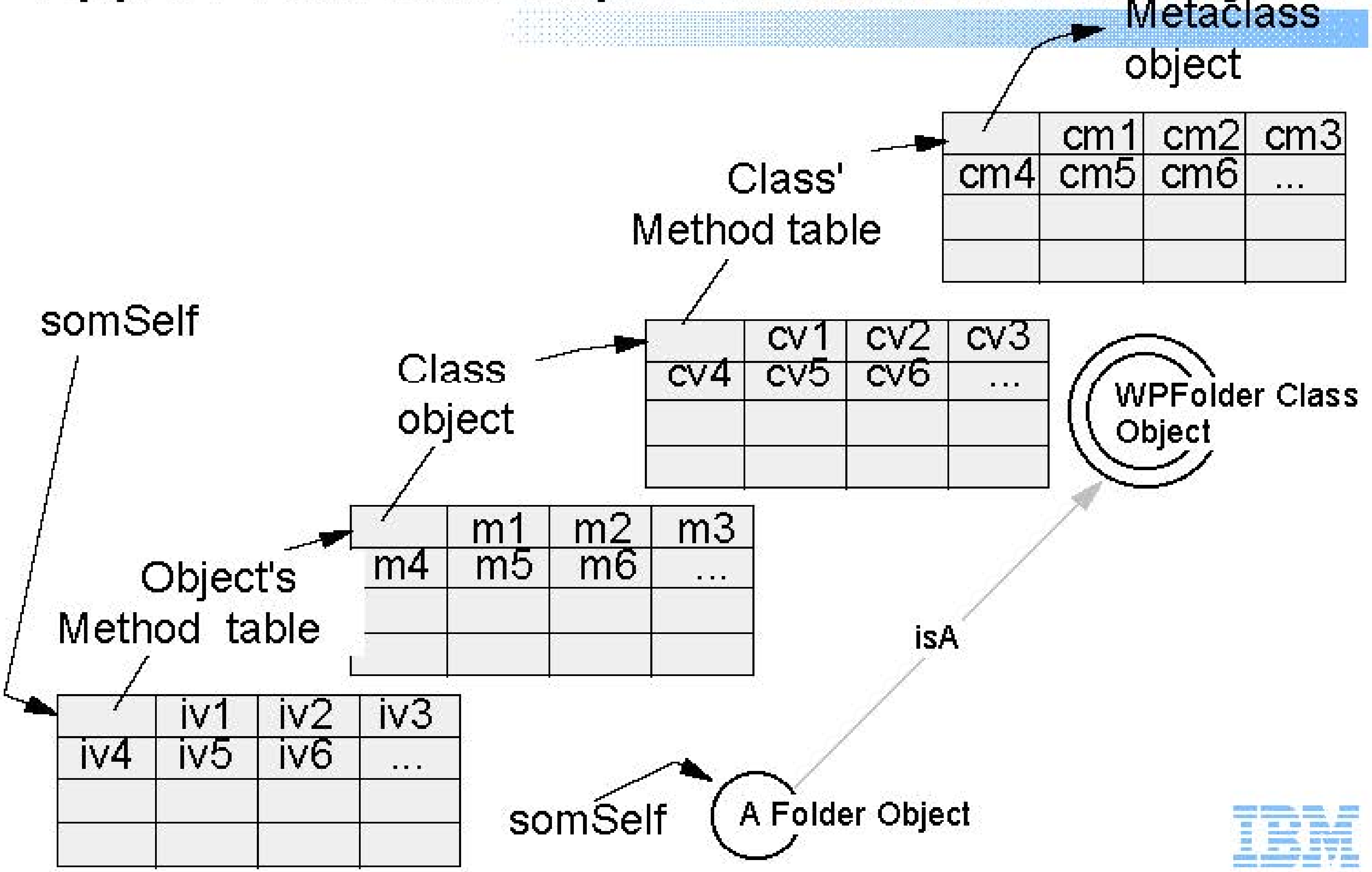
App A: Some new WPSH methods (cont'd)

- WPObject (cont'd)
  - wpclsInsertMultipleObjects
  - wpclsObjectFromHandle
  - wpclsRemoveObjects
  - wpclsQueryObjectFromFrame
  - wpclsQueryActiveDesktopHWND
  - wpclsQueryActiveDesktop
- WPFileSystem
  - wpQueryFilename
  - wpQueryDisk
  - wpQueryDateInfo
  - wpclsQueryObjectFromPath
- WPShadow
  - wpSetLinkToObject
- WPDisk
  - wpQueryDriveLockStatus
  - wpEjectDisk
  - wpLockDrive

App A: Some new WPSH methods (cont'd)

- WPFolder
  - wpModifyFldrFlags
  - wpAddToContent
  - wpDeleteFromContent
  - wpclsQueryOpenFolders
  - wpContainsFolders
  - wpclsQueryIconDataN

App B: Run-time Representation of SOMobjects*



Metaclass object

	cm1	cm2	cm3
cm4	cm5	cm6	...

Class' Method table

	cv1	cv2	cv3
cv4	cv5	cv6	...

WPFolder Class Object

somSelf

Class object

	m1	m2	m3
m4	m5	m6	...

Object's Method table

	iv1	iv2	iv3
iv4	iv5	iv6	...

isA

somSelf

A Folder Object