CUSTOM ORDER VISUALIZATION SYSTEM

by

Jennifer L. Zeis

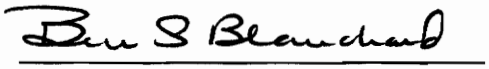
Project and Report submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Systems Engineering

APPROVED:



B.S. Blanchard, Chairman

H.A. Kurstedt

C.M. Waldorfer

April 26, 1994

Blacksburg, Virginia

CUSTOM ORDER VISUALIZATION SYSTEM

by

Jennifer L. Zeis

Committee Chairman: Dr. Benjamin S. Blanchard

Systems Engineering

(ABSTRACT)

The systems engineering life cycle can be applied to small projects as well as large ones. This paper applies the life cycle to a custom order visualization system for a small business. It addresses the definition of need, system operational requirements, maintenance concept, analysis, design and life cycle cost of the system. The system is prototyped in order to assess the feasibility of full implementation.

Table of Contents

List of Figures

1.0 Background

The proprietress of Joanie's, a small business that specializes in creating custom baby clothes by smocking designs onto them, desires to automate her business. The business started two years ago by displaying its wares at craft fairs. The business has expanded to include supplying three small boutiques. Joanie now fills orders from store buyers as well as from individuals. The patrons of these stores may not be familiar with the types of available designs. Joanie wants a system which will allow her customers to visualize their orders before they commit to the non-refundable purchase. The tool will be used by people with no computer experience and the storekeepers can not be expected to train each customer, so ease of use will be extremely important.

2.0 Definition of Need

Joanie's is a home-based business that provides the smocked product to several stores on either a commission or consignment basis. It is not cost effective to make the clothes in advance in anticipation of what the customers will want. Hence, the majority of sales consist of custom orders either by individuals or store buyers. Customers search through books of available designs known as plates and select the one they want. They also choose the clothing style and color. It is too costly to stock all fabric colors, but verbal descriptions rarely mean the same thing to both the customer and the supplier. When the business was selling to one store, the seamstress and the store buyer selected fabric and plates together. Now that Joanie's has grown to supply several stores, a more efficient purchasing method is needed. The seamstress fills 3-5 orders a month. She could handle up to twenty. Part of the difficulty in increasing sales stems from unfamiliarity with the product. Customers need a tool to visualize their $40-$100 order. As the business grows, tools are also needed for integrating customer trend, inventory, and accounting data.

To be available for the peak Christmas ordering season, the system needs to be completed by September 1, 1994.

The business employs only one person, so capital funds are limited to $5000. The business owner needs the

flexibility to utilize the same resources for multiple purposes and to limit specialized equipment.

The visualization tool takes first priority. An accounting system does exist, so a new accounting and inventory system has lower priority. Neither tool takes precedent over meeting operating expenses.

3.0 System Operational Requirements

The following represents an initial list of system operational requirements [1].

1.0 Mission Definition

1.1 The primary purpose of the system is to let customers and buyers visualize their orders in advance.

1.2 The secondary purpose is to provide an integrated method for accounting and inventory control.

1.3 The system shall use existing equipment where practical.

1.4 The system shall use only off-the-shelf hardware.

1.5 The system shall use off-the-shelf software where possible.

2.0 Performance and Physical Parameters

2.1 Size

2.1.1 The workstation shall not exceed 52 cm x 50 cm x 12 cm in dimension, less the monitor and keyboard.

2.1.2 The monitor shall not exceed 40 cm x 40 cm x 40 cm, with a usable screen size of at least 36 cm diagonal.

2.2 Input/Output

2.2.1 The workstation shall have a standard IBM-101 type keyboard.

2.2.2 The workstation shall have a MS-standard 2 or 3 button mouse.

2.2.3 The workstation shall accept input from a color scanner with at least 400 dpi.

2.2.4 The output shall be full (24 bit) color video with at least 1024 x 968 resolution.

2.2.5 The system shall support one color printer (at least 300 dpi resolution).

2.3 Capacity

2.3.1 The workstation shall have at least 8 MB of RAM storage.

2.3.2 The user workstation shall have at least 400 MB of hard disk space (< 15 ms).

2.3.3 The user workstation shall accept 3.5" (1.44 MB) and 5.25" (1.2 MB) floppy diskettes.

2.4 Performance

2.4.1 The system shall respond to all user requests within one minute. If the process time is greater than one minute, the user will be regularly informed of the progress of the process.

3.0 Usage Requirements

3.1 The system shall be capable of supporting the utilization shown in Figure 1 during the year.

3.2 The system shall be capable of short term bursts of 150% of projected utilization.

3.3 The system shall be capable of utilization from 9 am to 8 pm seven days a week.

4.0 Distribution

4.1 The initial system shall reside at one location.

4.2 The clothing design system shall be replicated at up to five sites.

4.3 The inventory and accounting systems shall remain at one site.

5.0 Operational Life Cycle

5.1 The workstations shall have an expected useful life of at least 5 years.

5.2 The software shall have an expected useful life of at least 5 years.

6.0 Effectiveness Measures

6.1 The system shall be available at least 94% of non-scheduled maintenance time.

6.2 The mean maintenance time shall not exceed 0.2% of operational time (MMH/OH).

6.3 The MTBM of the system shall be 2 years with an Mct of 2 hours.

6.4 The MDT shall not exceed 3 hours.

6.5 The MTBF of the system shall be 31 hours

6.6 The operational skill level shall be minimal.

6.7 The system acquisition cost shall not exceed $5000.

6.8 The annual operating cost shall not exceed $2500.

7.0 Environment

7.1 The system shall operate in a standard controlled office environment, as defined by OSHA, for averages and extremes of temperature, humidity, and lighting.

8.0 Human Computer Interface [2]

8.1 Initial Performance

8.1.1 The user shall be able to visualize a new design in 1 minute.

8.1.2 The user shall be able to change the design colors in 30 seconds.

8.2 Long Term Performance

8.2.1 The administrator shall be able to add pictures to the system in 15 minutes.

8.3 Learnability

8.3.1 A new user shall be able to learn the basic functionality in 2 minutes.

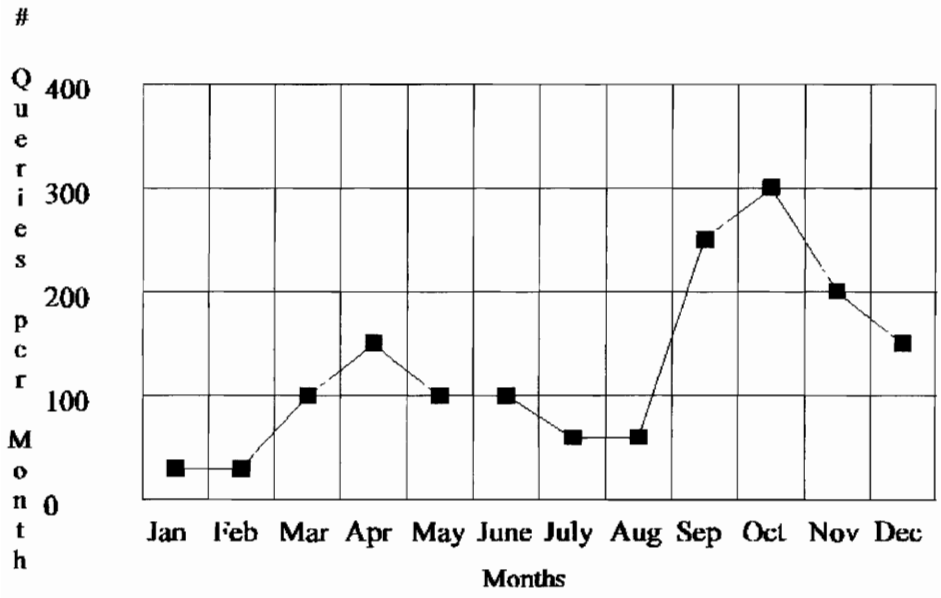


Figure 1: System Utilization Profile

4.0 Maintenance Concept

The system will be supported with two levels of maintenance: organizational and vendor [1]. Organizational maintenance shall be provided on-site by the system administrator. The system administrator position will be part time and be filled by a high school student who has the computer hardware skills. Two hours of training on the custom software will be required. The training will be performed by the software developer as part of the development cost. The individual vendors of the system's various components are responsible for vendor-level support at the vendors' servicing facilities.

4.1 Organizational Support

The organizational support will provide both preventive and corrective maintenance. Preventive maintenance activities include visual inspection of workstations and peripheral devices, replacement/replenishment of basic operating supplies, and performance of system backups. Corrective maintenance includes limited repair of equipment and installing upgrades of system software. If a repair is required beyond the capabilities of the administrator, he or she will be responsible for contacting the appropriate vendor.

4.2 Vendor Support

The system consists of off-the-shelf hardware and a combination of off-the-shelf and custom software. Failure of

off-the-shelf products shall be repaired or replaced by the appropriate vendor. The custom software shall be supported by the developer.

5.0 Systems Analysis

5.1 Definition of Problem

The need is to develop a system which will allow customers to preview their orders. The system will also provide for inventory and accounting control. The system needs to be implemented by September 1 at a maximum implementation cost of $5000.

5.2 Identification of Feasible Alternatives

Inventory and accounting data are currently handled by a Microsoft™ Works database application. A subset of possible designs are illustrated in a notebook. Ideally these two management tools would be automated and integrated. Various subsets of the long-term system are considered for the current project.

The following four alternatives exist:

1.) An integrated automated design, inventory, and accounting system

2.) An automated inventory and accounting system and a manual design system

3.) An automated design system and manual inventory and accounting system

4.) Use of the current automated inventory system as a separate entity from a new automated design system.

5.3 Evaluation Requirements

The crucial evaluation criteria for this system will be cost, time to implementation, and ease of use [1]. The system goal will be to increase the customer base.

5.4 Decision

Due to time and budget constraints, the fourth option is chosen. The first option would at least a year to implement and could not meet the September 1 deadline. The second option does not address the stated need of a flexible design system. The third option would be redundant in that an automated inventory system exists, so there is no reason to create a manual one. The fourth option which includes an automated design system has the greatest potential for increasing the customer base. The design system will be built as a stand-alone application, but it shall be designed so as not to preclude future integration with an inventory and accounting system.

6.0 Preliminary System Design

6.1 Functional Analysis

The custom design system can be systematically decomposed through functional analysis [1]. An initial operational functional flow is shown in Figures 2-9. A partial maintenance functional flow is illustrated in Figure 10.

6.2 Allocation of Requirements

The operational requirements are allocated to the components of the visualization system. Figure 11 illustrates the maintenance requirements allocation to two levels of detail [1].

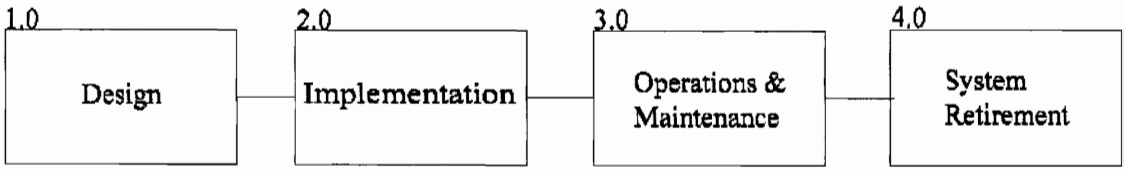


Figure 2: First-Level Functional Analysis

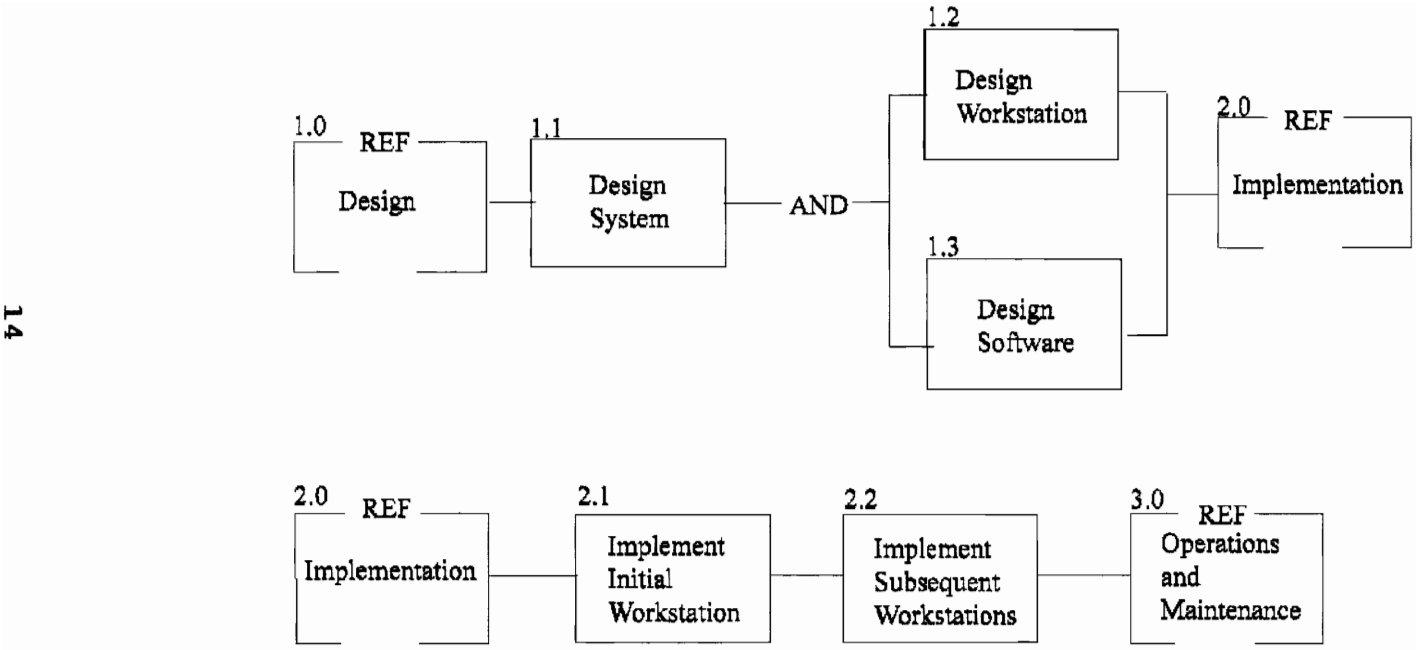


Figure 3: Second-Level Functional Analysis: Design and Implementation

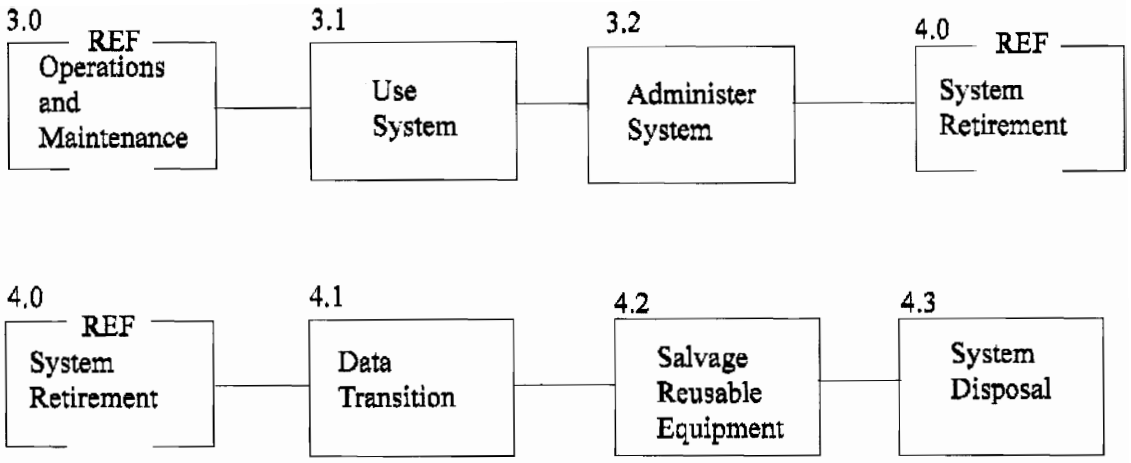


Figure 4: Second-Level Functional Analysis: O&M and System Retirement

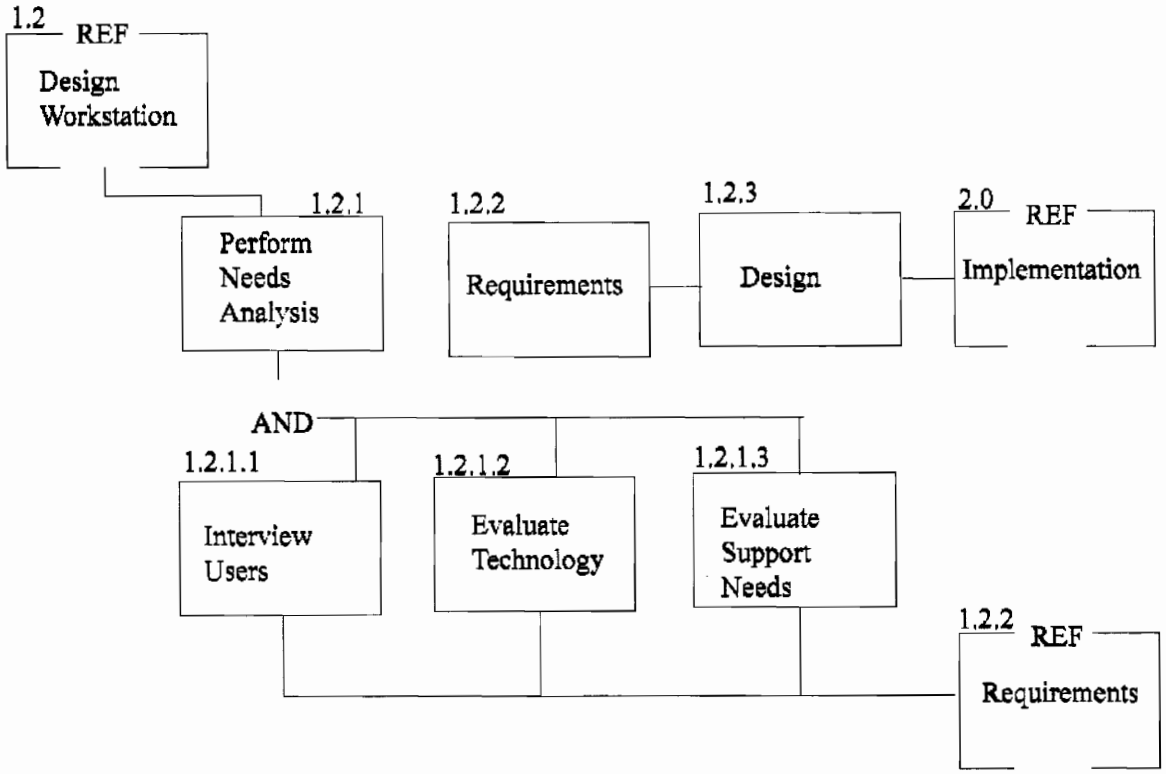


Figure 5: Flow for Workstation Design

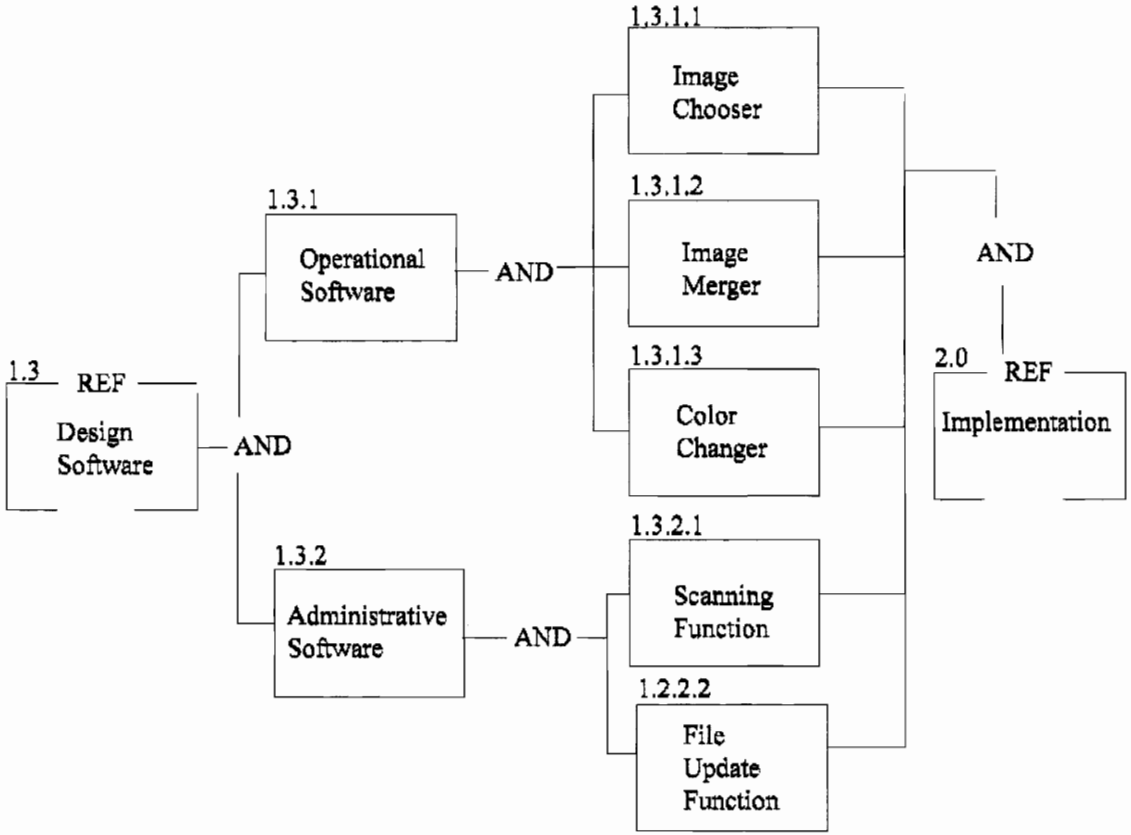


Figure 6: Flow for Software Design

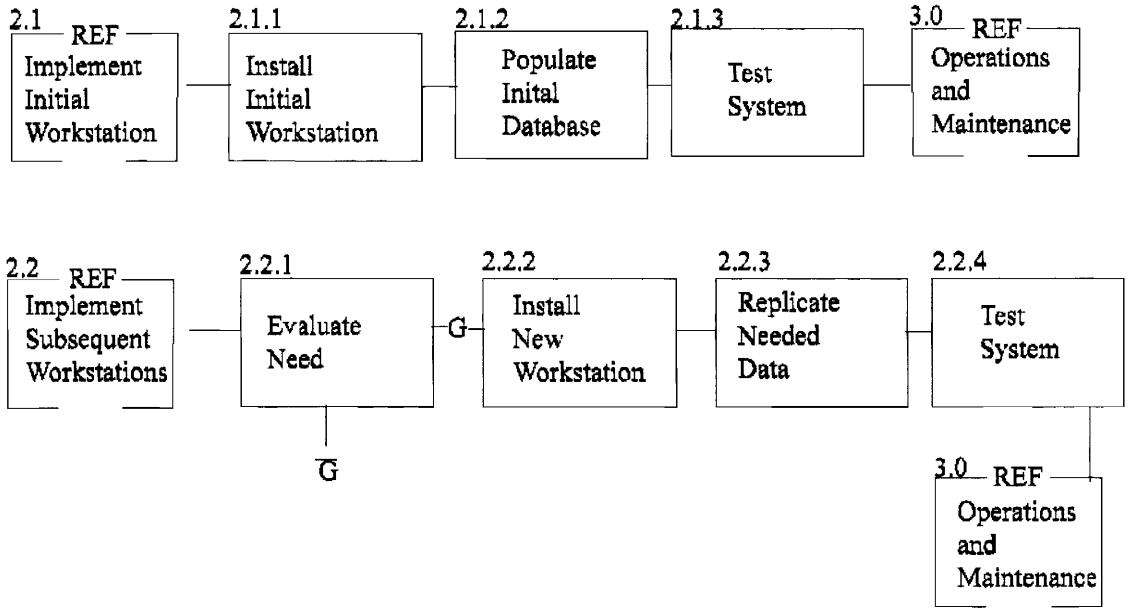


Figure 7: Implementation Functional Flow

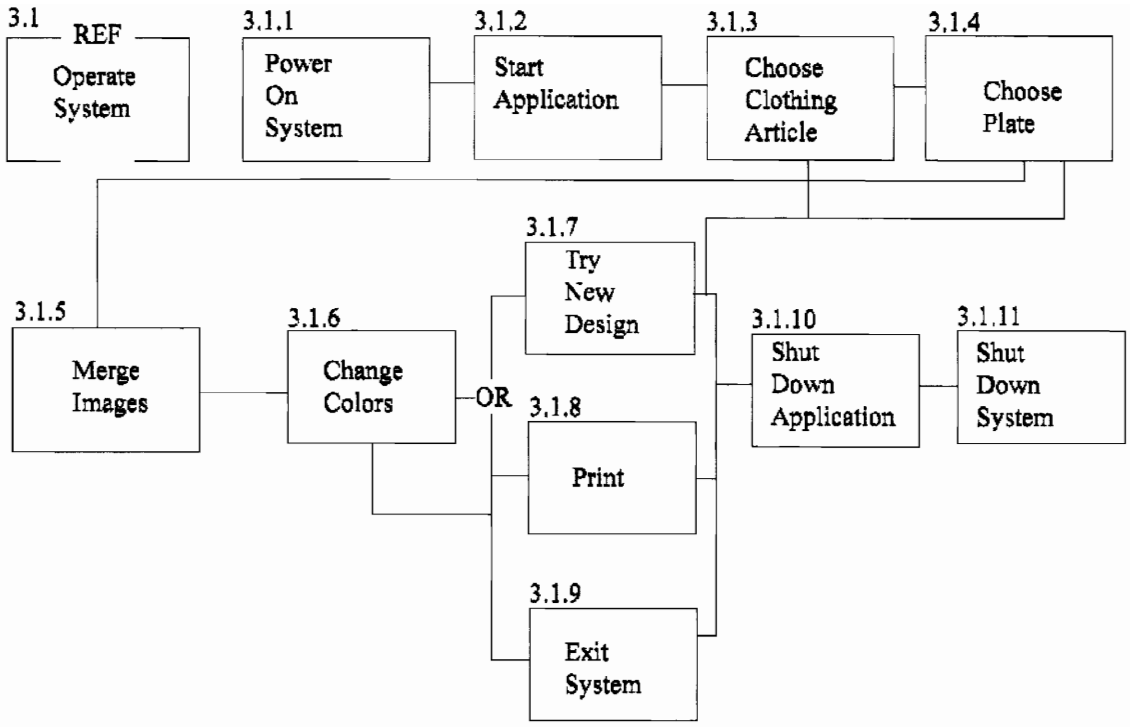


Figure 8: Operational Functional Flow

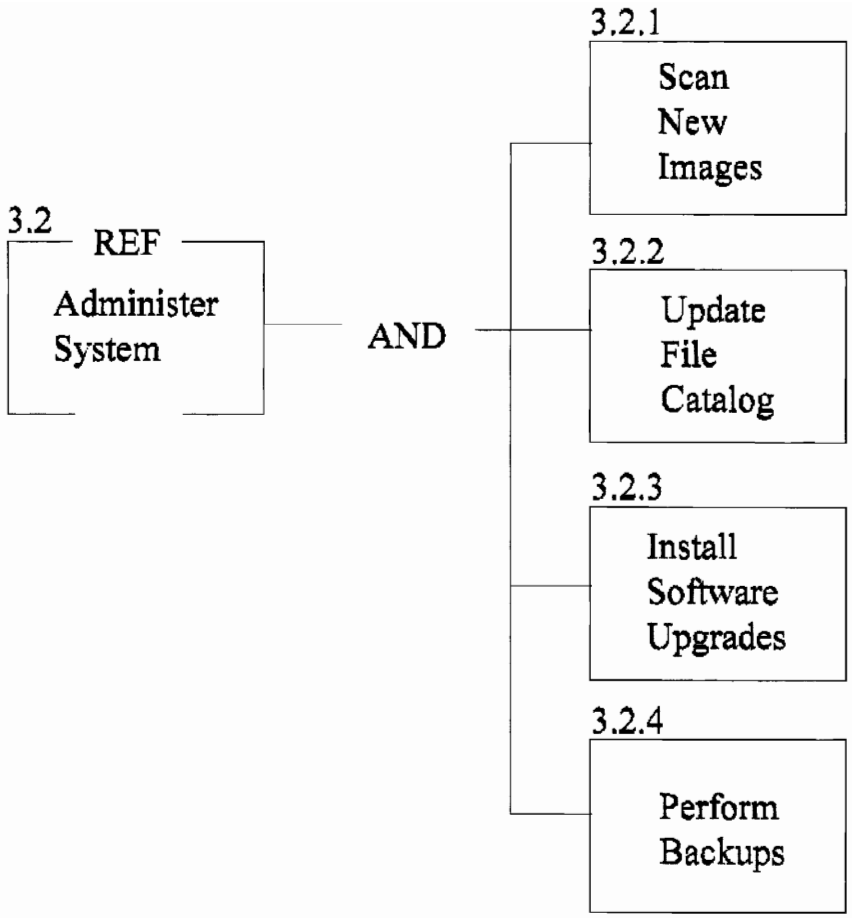


Figure 9: Administrative Functional Flow

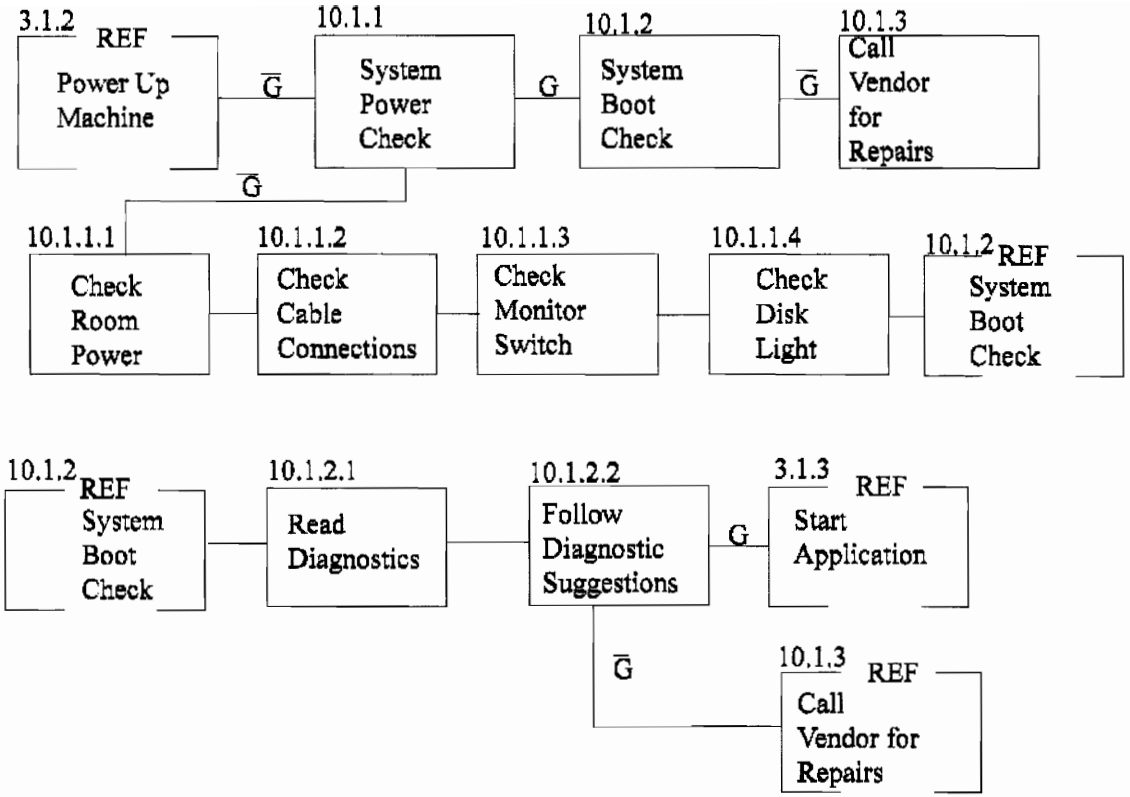


Figure 10: Maintenance Flow Diagram: Power-Up System

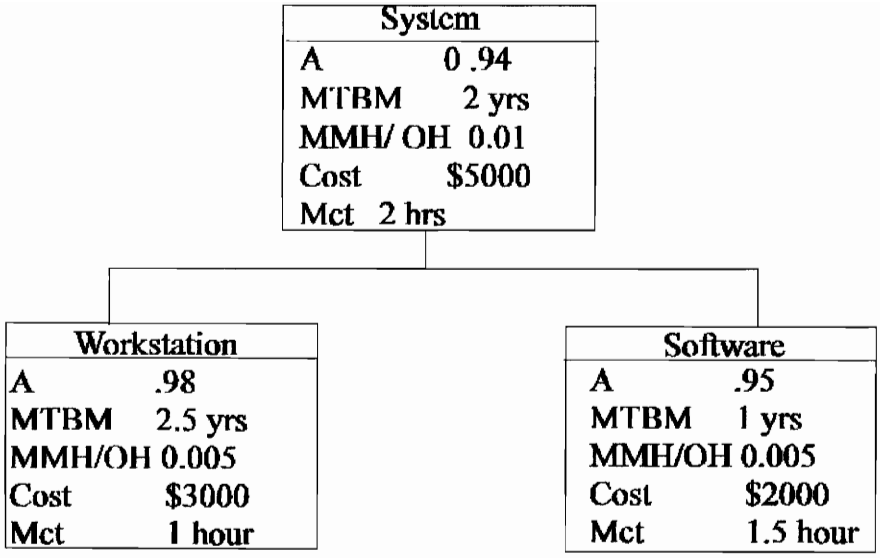


Figure 11: Allocation of Requirements

7.0 Detail Design and Development

7.1 Hardware Design

The hardware will consist of commercial off-the-shelf products. The products will be selected based on low purchase cost and comprehensive warranties which will reduce maintenance cost.

7.2 Software Design

The software will be custom designed to suit the needs of the business. The program will be written in C and C++ for the Microsoft Windows™ operating environment. It will conform to Windows™ interface design standards to facilitate ease of use [3]. It will be menu driven and include an on-line help system. A code generator will be used to quickly develop the user interface.

7.2.1 Operational Software

The operational software includes the system components that are available to the end user.

7.2.1.1 Main Module

The initial screen will be an empty window with a menu. The user will initially be able to select "choose dress", "choose plate", and "exit" from the "File" menu. If either "choose dress" or "choose plate" is selected, the system will enter the image chooser module. Once a new image is created, the user will be able to print or save the image. The seamstress will be able to use the saved image as the basis

for the order. These functions will be implemented using the standard Windows™ dialog boxes.

7.2.1.2 Image Chooser Module

The image chooser module will present the user with a list box containing the available images. After the user selects an image, it will be displayed in a child window in the main window [3]. A child window refers to a window which is displayed inside the main application window. The use of child windows allows the plate, dress, and merged images to be displayed simultaneously. The user will only be able to choose one dress and one plate at a time. After both images are chosen, the "merge pictures" and "change colors" menu options under "Custom" will become available. The system will enter either the image merger or color changer module based on the user's selection.

7.2.1.3 Image Merger Module

The image merger module contains the image processing routines which combine the images and display the new merged image. First, the plate bitmap is shrunk to a predetermined size to fit on the selected dress. The dress is copied to a new bitmap. The smaller plate is copied onto the new dress picture. This copy results in an image with harsh edges where the images meet. To reduce this effect, mean filtering is used [4], [5]. A 3x3 mask is slid along the border of the plate. The new values for the pixels are computed by taking

the average of the current pixel value and the values for the eight surrounding pixels. If each pixel was represented as a single value, the new value could be calculated using a single averaging equation. However, Windows™ bitmaps are stored as three components: red, green, and blue. Each of the components needs to be averaged separately to formulate the components for the new pixel value [3]. The mean filter results in smoothed edges between the two segments of the picture. After the composite image has been smoothed, it will be displayed in a new child window on the screen.

7.2.1.4 Color Changer Module

The customer will be able to change the colors in the merged image. The user will use the mouse to click on a pixel containing the color to be changed. The user will then select a new color from the standard Windows™ color selection dialog box. All pixels that match the selected one will be changed to the new color. To avoid dithering difficulties, only those colors that can be directly displayed on the screen will be available for selection [3].

7.2.2 Administrative Software

The administrative software will be used by the system administrator to update the picture selections. The administrative software will be divided into a scanning function and a file update function. As this software will be used by a more experienced user, the interface can be less

interactive. Initially, the administrative functions will be handled separately from the users' application.

7.2.2.1 Scanning Functions

Scanning will be accomplished using the software included with the scanner. New files will be scanned on a monthly basis to reflect the change of seasons and newly-available styles. The scanned images will be stored in the directory read by the user application. When multiple sites are operational, the same images will be replicated at each site.

7.2.2.2 File Update Function

Each dress image has a corresponding text file which contains the size and location of the rectangle into which to copy the selected plate. A new file is created for each new bitmap. It has the same name as the bitmap but with a .dat instead of a .bmp extension. The coordinates to enter can be found by reading the picture into a commercial paint package, selecting the target rectangle, and reading the coordinate values of the four corners. The data file can be created with any available text editor.

7.3 Prototype

Before the entire system is implemented, a prototype will be built. It will include some of the basic functionality but will not be as robust in terms of error checking as a finished application. A first version of the prototype screens is shown in Appendix A. This version allows the user to merge

images but does not include the ability to change colors. The administrative software is not included in the prototype. The flow diagrams and custom source code [3], [6], [7] used to create the prototype are included in Appendix B. Future implementation of the prototype would involve adding the missing functionality, adding error checking capability and performing system tests.

8.0 Life Cycle Cost

The life cycle costs were analyzed to calculate the system costs for 5 years at 5% inflation [1]. The life cycle cost includes the factors shown in Figure 12.

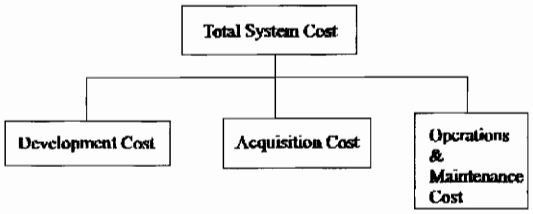


Figure 12: Components of Life Cycle Cost

The $2000 development costs assume a 100 hour development at $20/hr. The acquisition costs include:

Intel 4DX2-50 8 MB RAM 424 MB 13 ms hard drive 5.25" and 3.5" floppy drives MS DOS 6.2 and Windows™ for Workgroups	$1995.00
Color ink jet printer	$ 489.00
Color print cartridge	$ 29.00
Black print cartridge	$ 25.00
Tape Backup (250 MB)	$ 130.00
Color Scanner	$ 395.00
Total	$3063.00

The recurring operations and maintenance costs include:

Item	Cost Each	# per year	Annual Cost
Paper	$20.00 (ream)	12	$ 240.00
Ink (Color)	$29.00	12	$ 348.00
Ink (Black)	$25.00	12	$ 300.00
Tapes	$18.00	12	$ 216.00
Admin Salary	$10.00 (hour)	250 hrs	$2500.00
Total			$3604.00

The total future system cost at 5% inflation would be:

Year	Costs(1994$)	f/p/5/n	Future Cost
0	$5000	1.000	$ 5000.00
1	$3604	1.050	$ 3784.20
2	$3604	1.103	$ 3975.21
3	$3604	1.158	$ 4173.43
4	$3604	1.216	$ 4382.46
5	$3604	1.276	$ 4598.70
			$25,914.00

9.0 Conclusions

The prototype was demonstrated to the customer. She stated that while the concept is useful the total life cycle cost of $25,914.00 is extremely expensive for a small business. If the system meets the defined need, it would result in an 180 additional orders a year. The profit on each item is $20. Therefore, the design system would increase revenue by $3600 a year. The annual system cost is found to be $3604. Therefore, even in a best case scenario the system will only increase revenue enough to offset the annual cost. The acquisition cost would result in a reduction of profits. In order to be profitable, the annual cost would need to be reduced to $2500. Therefore, the following recommendation is made. The prototype will be expanded to become a working model. For the present, it will be run on the proprietress' home machine. She will be able to demonstrate the system to the stores' buyers. If they think the system has potential, they can contribute to the acquisition and development costs. Additionally, the system costs should be reviewed every six months to determine if falling hardware prices increase the system feasibility.

References

[1] B.S. Blanchard and W. J. Fabrycky, Systems Engineering and Analysis, Englewood Cliffs, NJ: Prentice-Hall Inc., 1990.

[2] D. Hix and H. R. Hartson, Developing User Interfaces/ Ensuring Usability through Product and Process, New York: John Wiley & Sons, Inc., 1993.

[3] C. Petzold, Programming Windows 3.1, 3rd ed, Redmond, WA: Microsoft Press, 1992.

[4] J. Lim, Two Dimensional Signal and Image Processing, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1990.

[5] W. K. Pratt, Digital Image Processing, 2nd ed, New York: John Wiley & Sons, Inc., 1991.

[6] S. Holzner, Borland C++ Windows Programming, 3rd ed. Indianapolis, IN: Brady Publishing, 1994.

[7] P. Perry, Turbo C++ for Windows: Programming for Beginners, Carmel, IN: Sams Publishing, 1993.

[7] W. K. Pratt, Digital Image Processing, 2nd ed, New York: John Wiley & Sons, Inc., 1991.

Glossary

Child Window
A window contained within another window in a user interface

Dithering
The combining of pixel patterns to create a color which is not inherently supported by the display device.

Inherent Availability (A_i)
The probability that the system will be available when used in an ideal support environment

Initial Performance
A measure of a user's performance the first time a system is used.

Learnability
A measure of how quickly and easily a user can learn a system.

Long Term Performance
A measure of user performance over an extended period of time.

Main Window
The window on a graphical interface which contains the main functionality of the application.

Mean Corrective Maintenance Time (Mct)
The average cycle time to repair a system or component

Mean Down Time (MDT)
The average total time required to repair a system to full operating status or to maintain full operating status.

Mean Filtering
An image processing technique for smoothing an image where pixels are assigned a value equivalent to the average values of the nine surrounding pixels.

Mean Maintenance Time (MMH/OH)
The proportion of average maintenance hours to operating hours for the system.

Mean Time Between Failure (MTBF)
The average time a system operates before a failure occurs.

Mean Time Between Maintenance (MTBM)
The average time a system operates before either preventive or corrective maintenance is required.

Appendix A: Prototype Screens

Figure 13: Main Screen and Dress Window

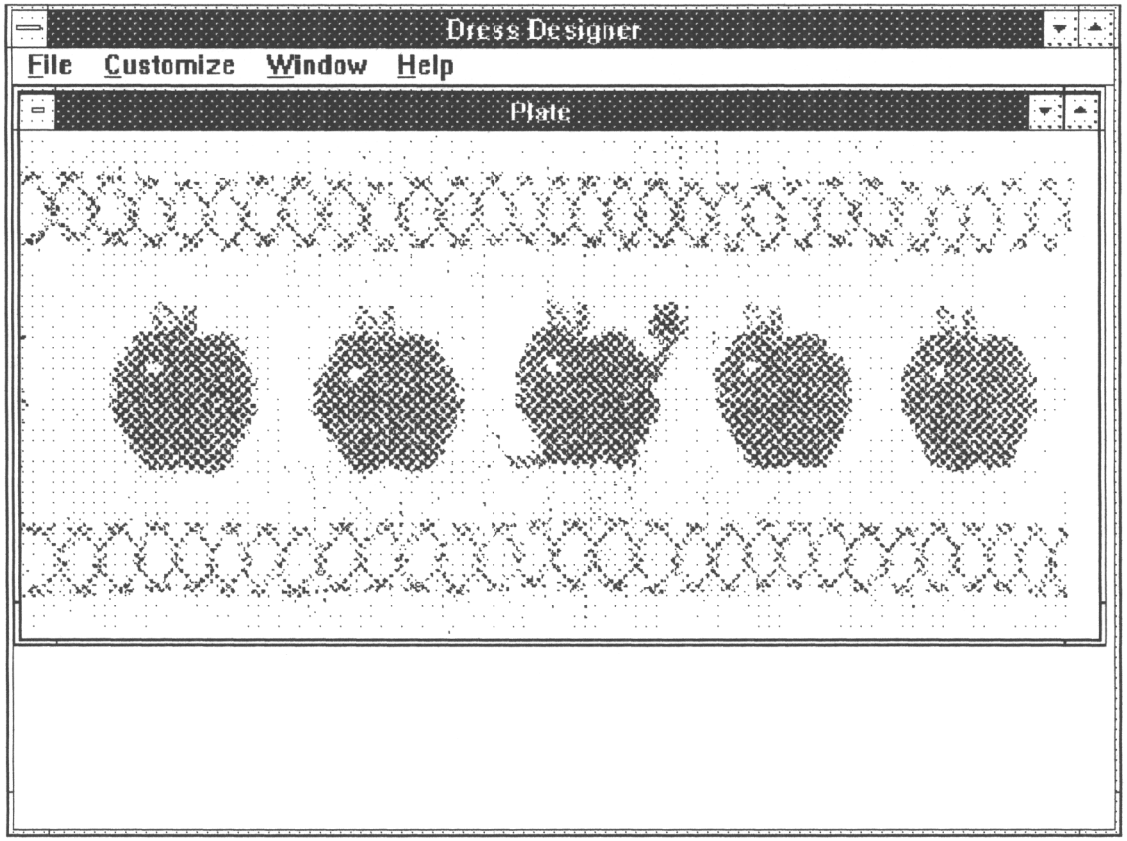


Figure 14: Plate Window

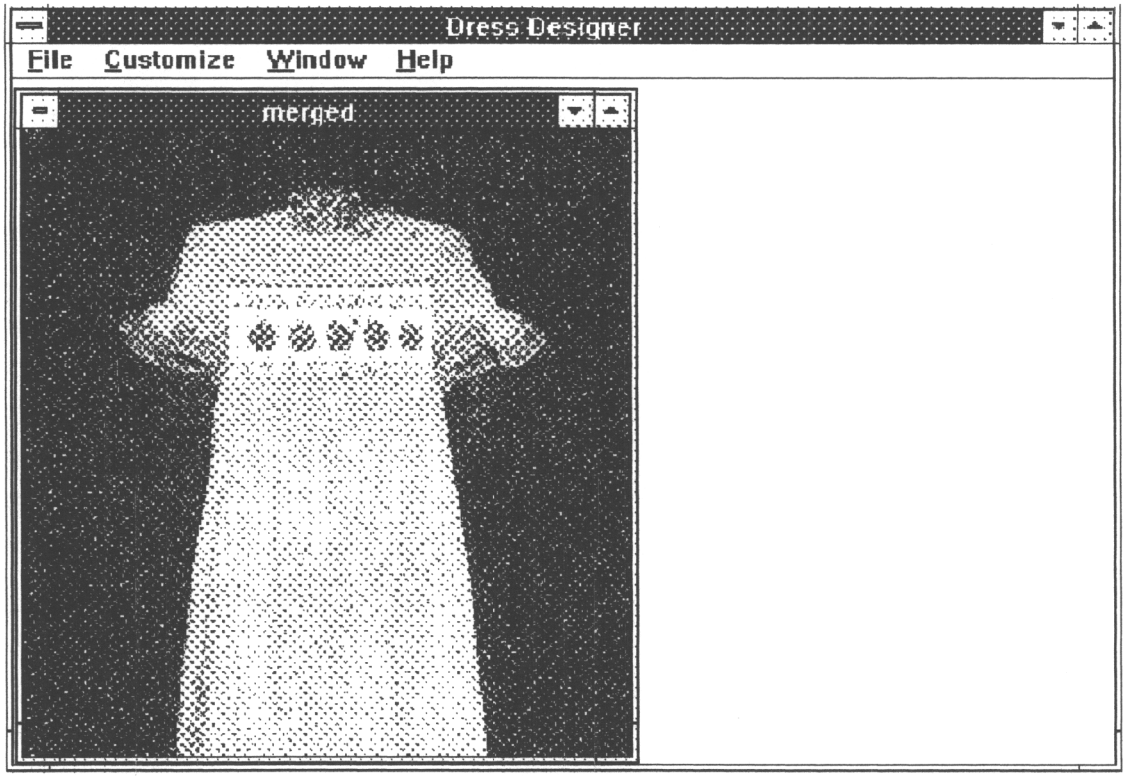


Figure 15: Final Design

Appendix B: Prototype Source Code

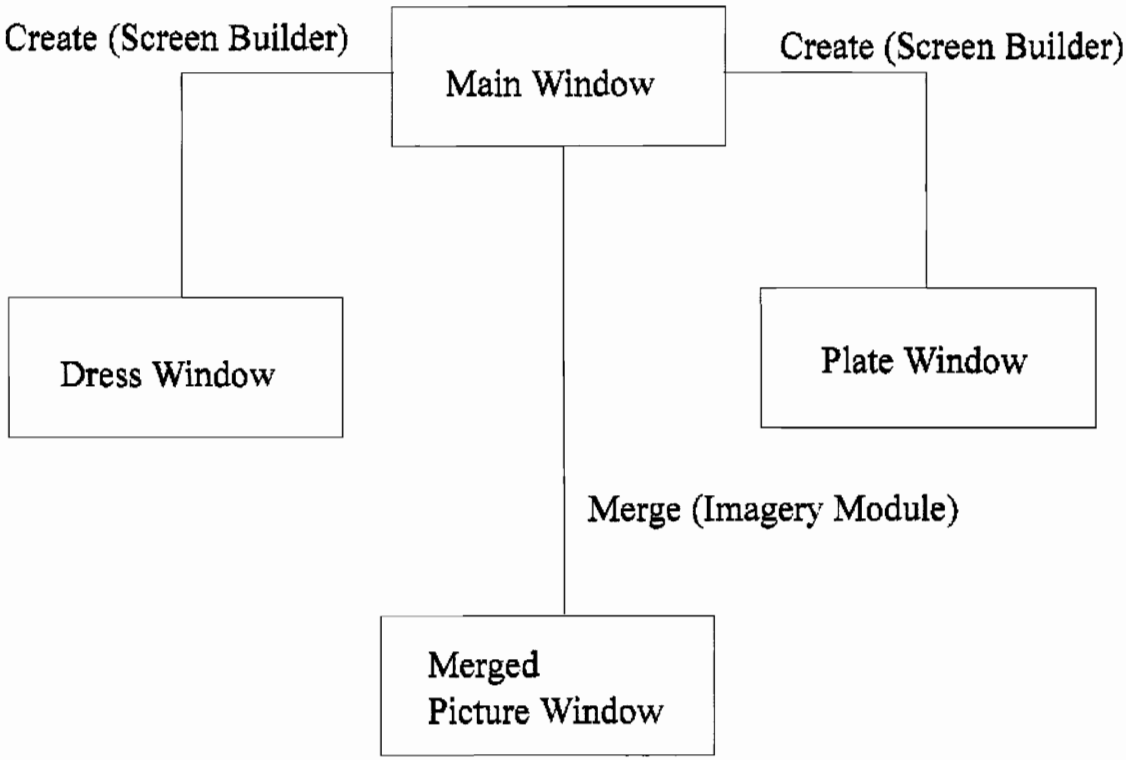


Figure 16: Windowing Structure

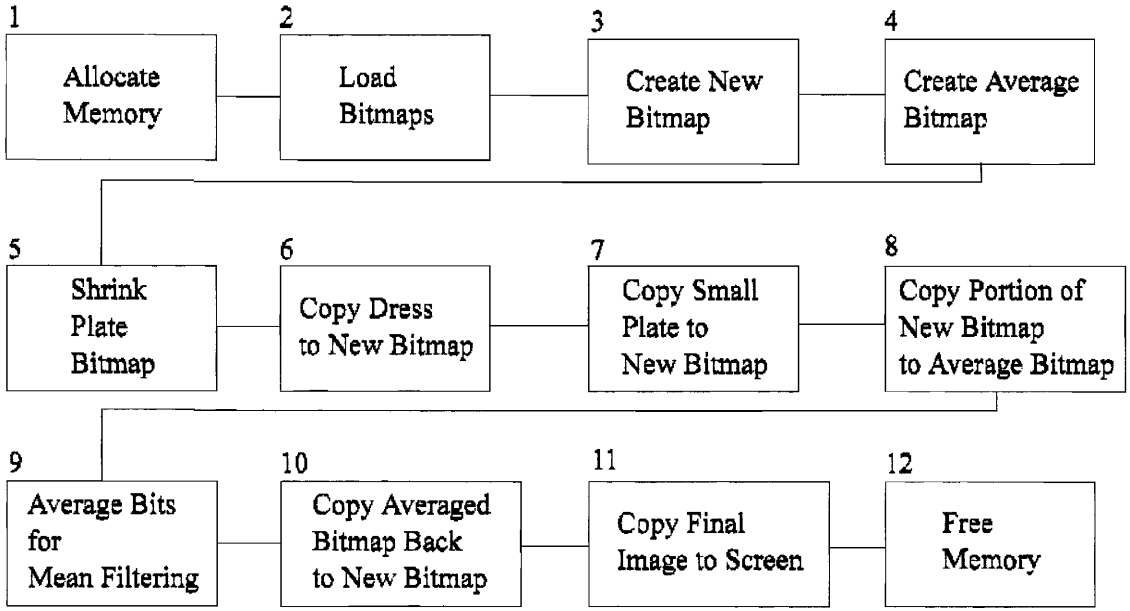


Figure 17: Image Merger Module Flow

```
/* This module includes the image processing routines */

#include "AFXWIN.H"
#include <fltpnt.h>

void merger (CWnd* myWnd)
{       // Insert your code here
//static BITMAP bitmap = {0, 20, 5, 4, 1, 1};
//    static BYTE byBits[] = {0x51, 0x77, 0x10, 0x00,
//                            0x57, 0x77, 0x50, 0x00,
//                            0x13, 0x77, 0x50, 0x00,
//                            0x57, 0x77, 0x50, 0x00,
//                            0x51, 0x11, 0x10, 0x00};

    BITMAP bm, bm2, bm3;
    HDC hdc, hdcMem1, hdcMem2, hdcMem3, hdcMem4, hdcMem5;
    CDC* pdc = 0;
    DWORD dwSize;
    POINT ptSize, ptOrg;
    HBITMAP hBitmapDress, hBitmapDesign, hBitmapPlate,
         hBitmapShrunk,
            hBitmapAverage;
    PAINTSTRUCT ps;
    RECT rect;
    LONG dwCount;
    char _far *lpBits;
    char szBuffer[256];
    int i;
    long rowLength;
    float a;

     // Insert your code here


    pdc = myWnd->BeginPaint(&ps);

    hBitmapPlate = ::LoadBitmap(hBMPInst,"PLATE16");
    hBitmapDress = ::LoadBitmap(hBMPInst,"DRESS16");



    GetObject (hBitmapPlate, sizeof(BITMAP), (LPSTR) &bm);
    hBitmapShrunk   =   CreateBitmap(106,   54,   bm.bmPlanes,
bm.bmBitsPixel, NULL);

    GetObject (hBitmapDress, sizeof(BITMAP), (LPSTR) &bm2);

    hBitmapDesign = CreateBitmap (bm2.bmWidth, bm2.bmHeight,
```

```
                bm2.bmPlanes, bm2.bmBitsPixel, NULL);

    hBitmapAverage = CreateBitmap (110, 58, bm.bmPlanes,
bm.bmBitsPixel, NULL);

    hdcMem1 = CreateCompatibleDC (pdc->m_hDC);
    hdcMem2 = CreateCompatibleDC (pdc->m_hDC);
    hdcMem3 = CreateCompatibleDC (pdc->m_hDC);
    hdcMem4 = CreateCompatibleDC (pdc->m_hDC);
    hdcMem5 = CreateCompatibleDC (pdc->m_hDC);

    SelectObject (hdcMem1, hBitmapPlate);
    SelectObject (hdcMem2, hBitmapShrunk);
    SelectObject (hdcMem3, hBitmapDress);
    SelectObject (hdcMem4, hBitmapDesign);
    SelectObject (hdcMem5, hBitmapAverage);
// Shrink Plate
    SetStretchBltMode(hdcMem2, COLORONCOLOR);

    StretchBlt (hdcMem2, 0, 0, 106, 54,
          hdcMem1, 0, 0, bm.bmWidth, bm.bmHeight, SRCCOPY);


// Copy Dress to New Bitmap

    BitBlt (hdcMem4, 0, 0, bm2.bmWidth, bm2.bmHeight,
         hdcMem3, 0, 0, SRCCOPY);

// Copy small plate to new bitmap

    BitBlt (hdcMem4, 108, 87, 106, 50,
         hdcMem2, 0, 0, SRCCOPY);

 // Copy portion of new bitmap to average bitmap
//   BitBlt (hdcMem5, 0, 0, 110, 58,
//      hdcMem4, 106, 81, MERGECOPY);

// Get character array of bits

   GetObject (hBitmapAverage, sizeof(BITMAP), &bm3);
   dwCount = (DWORD) bm3.bmWidthBytes * bm3.bmHeight *
bm3.bmPlanes;
   lpBits = new char[dwCount];

   if ((lpBits != 0) && (dwCount < 64000)) {
   GetBitmapBits (hBitmapAverage, dwCount, lpBits);

   // Calculate Average
    rowLength = 54;
```

```
   for (i=55; i < 109; i++) {
     a = lpBits[i];
                                          a                          =
(1/9)*((lpBits[i-rowLength])+(lpBits[i+rowLength])+(lpBits[i
-1])
        +(lpBits[i+1])+(lpBits[i-rowLength+1])
            +(lpBits[i-rowLength-1])+(lpBits[i+rowLength-1])

            +(lpBits[i+rowLength+1]));
     lpBits[i] = truncf(a);
      }


   // Set bits back to averaged bitmap

//   SetBitmapBits (hBitmapAverage, dwCount, lpBits);

   // Copy averaged bitmap back to new bitmap

 //  BitBlt (hdcMem4, 106, 81, 110, 58,
//      hdcMem5, 0, 0, SRCCOPY);

   // Copy final image to screen

   BitBlt (pdc->m_hDC, 0, 0, bm2.bmWidth, bm2.bmHeight,
         hdcMem4, 0, 0, SRCCOPY);
   }
   else
   pdc->TextOut(0,0,"Allocation error", 16);


    DeleteDC (hdcMem1);
    DeleteDC (hdcMem2);
    DeleteDC (hdcMem3);
    DeleteDC (hdcMem4);
    DeleteDC (hdcMem5);

    DeleteObject (hBitmapShrunk);
    DeleteObject (hBitmapDesign);
    DeleteObject (hBitmapAverage);

    delete[] lpBits;
    lpBits = 0;

     myWnd->EndPaint(&ps);
}
```

```
// Filename: USERCODE.CPP

// "DESIGNER" Generated by Visual Programmer.

// Author:


//
/                                                                      /
************************************************************
***********
// Code in this file is initially generated by the Switch-It
Module.
// This file contains functions you can change
// to provide whatever functionality you require.
//
//
//
// For more information,
//  see  the  section  "How  code  is  generated"  in  the
documentation.
//
/                                                                      /
************************************************************
***********
//

#include "AFXWIN.H"
#include "DESIGNER.H"

WMPDEBUG

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif


#include "USERCODE.WMC"




/                                                                      /
************************************************************

//   C Startup function for window MDICHILD
/                                                                      /
```

```
**************************************************************


CWnd* BLD_MDICHILDCreateWnd(CWnd *pAParent)
    {
    Cwm_MDICHILDWnd *pMDICHILDWnd;

    // Create the object
    pMDICHILDWnd = new Cwm_MDICHILDWnd();
    if (!pMDICHILDWnd)
        return NULL;

    // Create the window
    pMDICHILDWnd->SimInitWindow(pAParent);

    return pMDICHILDWnd;
    }



/                                                             /
**************************************************************

// Member Functions for Window: Cwm_MDICHILDWnd
// Base Class                 : Cwm_MDICHILDBaseWnd
// Derived from MFC Class     : CWnd
/                                                             /
**************************************************************


Cwm_MDICHILDWnd :: Cwm_MDICHILDWnd()
    : Cwm_MDICHILDBaseWnd()
{
}


Cwm_MDICHILDWnd :: ~Cwm_MDICHILDWnd()
{
}


IMPLEMENT_DYNCREATE(Cwm_MDICHILDWnd,Cwm_MDICHILDBaseWnd)

BEGIN_MESSAGE_MAP(Cwm_MDICHILDWnd,Cwm_MDICHILDBaseWnd)

    // YOU CAN ADD YOUR OWN CODE HERE

    //{{SIM_MSG_MAP(Cwm_MDICHILDWnd)
    // DO NOT EDIT what you see in this block of generated
```

```
code.
    //}}SIM_MSG_MAP(Cwm_MDICHILDWnd)

END_MESSAGE_MAP()


//{{SIM_MSG_BODY(Cwm_MDICHILDWnd)
// Message Handler Functions for you to edit will come here.









//}}SIM_MSG_BODY(Cwm_MDICHILDWnd)



/                                                                       /
************************************************************************

// Cwm_MDICHILDWnd diagnostics

#ifdef _DEBUG
void Cwm_MDICHILDWnd::AssertValid() const
{
    Cwm_MDICHILDBaseWnd::AssertValid();
}

void Cwm_MDICHILDWnd::Dump(CDumpContext& dc) const
{
    Cwm_MDICHILDBaseWnd::Dump(dc);
}

#endif //_DEBUG



/                                                                       /
************************************************************************
```

```
//   C Startup function for window Plate
/                                                                       /
************************************************************************


CWnd* BLD_PlateCreateWnd(CWnd *pAParent)
    {
    Cwm_PlateWnd *pPlateWnd;

    // Create the object
    pPlateWnd = new Cwm_PlateWnd();
    if (!pPlateWnd)
        return NULL;

    // Create the window
    pPlateWnd->SimInitWindow(pAParent);

    return pPlateWnd;
    }



/                                                                       /
************************************************************************

// Member Functions for Window: Cwm_PlateWnd
// Base Class                 : Cwm_PlateBaseWnd
// Derived from MFC Class     : CWnd
/                                                                       /
************************************************************************


Cwm_PlateWnd :: Cwm_PlateWnd()
    : Cwm_PlateBaseWnd()
{
}


Cwm_PlateWnd :: ~Cwm_PlateWnd()
{
}


IMPLEMENT_DYNCREATE(Cwm_PlateWnd,Cwm_PlateBaseWnd)

BEGIN_MESSAGE_MAP(Cwm_PlateWnd,Cwm_PlateBaseWnd)

    // YOU CAN ADD YOUR OWN CODE HERE
```

```
    //{{SIM_MSG_MAP(Cwm_PlateWnd)
    // DO NOT EDIT what you see in this block of generated
code.
    //}}SIM_MSG_MAP(Cwm_PlateWnd)

END_MESSAGE_MAP()


//{{SIM_MSG_BODY(Cwm_PlateWnd)
// Message Handler Functions for you to edit will come here.









//}}SIM_MSG_BODY(Cwm_PlateWnd)



/                                                                    /
**********************************************************************

// Cwm_PlateWnd diagnostics

#ifdef _DEBUG
void Cwm_PlateWnd::AssertValid() const
{
    Cwm_PlateBaseWnd::AssertValid();
}

void Cwm_PlateWnd::Dump(CDumpContext& dc) const
{
    Cwm_PlateBaseWnd::Dump(dc);
}

#endif //_DEBUG
```

```
/                                                                       /
************************************************************************

//   C Startup function for window Design
/                                                                       /
************************************************************************


CWnd* BLD_DesignCreateWnd(CWnd *pAParent)
    {
    Cwm_DesignWnd *pDesignWnd;

    // Create the object
    pDesignWnd = new Cwm_DesignWnd();
    if (!pDesignWnd)
        return NULL;

    // Create the window
    pDesignWnd->SimInitWindow(pAParent);


     return pDesignWnd;
    }



/                                                                       /
************************************************************************

// Member Functions for Window: Cwm_DesignWnd
// Base Class                 : Cwm_DesignBaseWnd
// Derived from MFC Class     : CWnd
/                                                                       /
************************************************************************


Cwm_DesignWnd :: Cwm_DesignWnd()
    : Cwm_DesignBaseWnd()
{
}


Cwm_DesignWnd :: ~Cwm_DesignWnd()
{
}


IMPLEMENT_DYNCREATE(Cwm_DesignWnd,Cwm_DesignBaseWnd)
```

```
BEGIN_MESSAGE_MAP(Cwm_DesignWnd,Cwm_DesignBaseWnd)

    // YOU CAN ADD YOUR OWN CODE HERE

    //{{SIM_MSG_MAP(Cwm_DesignWnd)
    // DO NOT EDIT what you see in this block of generated
code.
    //}}SIM_MSG_MAP(Cwm_DesignWnd)

END_MESSAGE_MAP()


//{{SIM_MSG_BODY(Cwm_DesignWnd)
// Message Handler Functions for you to edit will come here.









//}}SIM_MSG_BODY(Cwm_DesignWnd)



/                                                                    /
*********************************************************************

// Cwm_DesignWnd diagnostics

#ifdef _DEBUG
void Cwm_DesignWnd::AssertValid() const
{
    Cwm_DesignBaseWnd::AssertValid();
}

void Cwm_DesignWnd::Dump(CDumpContext& dc) const
{
    Cwm_DesignBaseWnd::Dump(dc);
}
```

```
#endif //_DEBUG


//  Function for MDI windows
BOOL BLD_CascadeSendMDI(CWnd *pWnd)
{
    return BLDSendMDIMessage(pWnd,WM_MDICASCADE,0);
}


//  Function for MDI windows
BOOL BLD_TileSendMDI(CWnd *pWnd)
{
    return BLDSendMDIMessage(pWnd,WM_MDITILE,0);
}


//  Function for MDI windows
BOOL BLD_ArrangeIconsSendMDI(CWnd *pWnd)
{
    return BLDSendMDIMessage(pWnd,WM_MDIICONARRANGE,0);
}


//  Function for MDI windows
BOOL BLD_CloseActiveSendMDI(CWnd *pWnd)
{
    return BLDSendMDIMessage(pWnd,WM_MDIGETACTIVE,0);
}



/                                                              /
**************************************************************

// Class definition for Class: Cwm_ChooseDressDlg
// Base Class                : Cwm_ChooseDressBaseDlg
// Derived from MFC Class    : CDialog
/                                                              /
**************************************************************


char  dressChoices[2][20] = {"Bishop Dress",
               "Bubble Dress",};

class Cwm_ChooseDressDlg : public Cwm_ChooseDressBaseDlg
{
```

```
public:
     Cwm_ChooseDressDlg(LPCSTR   lpszTemplateName,CWnd
*pParentWnd);
    Cwm_ChooseDressDlg() : Cwm_ChooseDressBaseDlg() {};

#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

    virtual BOOL OnInitDialog();
    virtual void OnOK();
    virtual void OnCancel();

    // YOU CAN ADD YOUR OWN CODE HERE


protected:
    //{{SIM_MSG_PROTO(Cwm_ChooseDressDlg)
    // DO NOT EDIT what you see in this block of generated
code.
    void ucInitdialogDress_Li(void);
    //}}SIM_MSG_PROTO(Cwm_ChooseDressDlg)

protected:
    virtual  void  DoDataExchange(CDataExchange*  pDX);     //
DDX/DDV support

    DECLARE_DYNCREATE(Cwm_ChooseDressDlg)

    DECLARE_MESSAGE_MAP()

};


/                                                           /
************************************************************

//   C Startup function for modal dialogbox DRESSLIST
/                                                           /
************************************************************


int BLD_ChooseDressDlgFunc(CWnd *pWnd)
    {
    int iRet;
    Cwm_ChooseDressDlg TheDlg ("DRESSLIST",pWnd);

    iRet=TheDlg.DoModal();
```

```
    if( -1 == iRet )
        {
         BLDDisplayMessage    (::GetActiveWindow    (),
BLD_CannotCreate,
            "DRESSLIST", MB_OK | MB_ICONHAND);
        }
    return iRet;
    }



/                                                                    /
**********************************************************************

// Member Functions for Class: Cwm_ChooseDressDlg
// Base Class                : Cwm_ChooseDressBaseDlg
// Derived from MFC Class    : CDialog
/                                                                    /
**********************************************************************


Cwm_ChooseDressDlg::Cwm_ChooseDressDlg(LPCSTR
lpszTemplateName,CWnd *pParentWnd)
  : Cwm_ChooseDressBaseDlg(lpszTemplateName,pParentWnd)
{
}



BOOL Cwm_ChooseDressDlg::OnInitDialog()
{
    return Cwm_ChooseDressBaseDlg::OnInitDialog();
}



void Cwm_ChooseDressDlg::OnOK()
{
    Cwm_ChooseDressBaseDlg::OnOK();
}



void Cwm_ChooseDressDlg::OnCancel()
{
    Cwm_ChooseDressBaseDlg::OnCancel();
}
```

```
IMPLEMENT_DYNCREATE(Cwm_ChooseDressDlg,
Cwm_ChooseDressBaseDlg)

BEGIN_MESSAGE_MAP(Cwm_ChooseDressDlg,Cwm_ChooseDressBaseDlg)


    // YOU CAN ADD YOUR OWN CODE HERE

    //{{SIM_MSG_MAP(Cwm_ChooseDressDlg)
    // DO NOT EDIT what you see in this block of generated
code.
    //}}SIM_MSG_MAP(Cwm_ChooseDressDlg)

END_MESSAGE_MAP()


//{{SIM_MSG_BODY(Cwm_ChooseDressDlg)
// Message Handler Functions for you to edit will come here.




// CONTROL NOTIFICATION
void Cwm_ChooseDressDlg::ucInitdialogDress_Li( )
     {
     // Insert your code here
     int i;
       for(i=0;i<=1;i++)
          SendDlgItemMessage(ID_Dress_List,LB_ADDSTRING,  0,
(LONG)dressChoices[i]);

}

//}}SIM_MSG_BODY(Cwm_ChooseDressDlg)



void  Cwm_ChooseDressDlg::DoDataExchange(CDataExchange*  pDX)

{
    Cwm_ChooseDressBaseDlg::DoDataExchange(pDX);

    // YOU CAN ADD YOUR OWN CODE HERE


    //{{SIM_DATA_MAP(Cwm_ChooseDressDlg)
    // DO NOT EDIT what you see in this block of generated
```

```
code.

    //}}SIM_DATA_MAP
}



/                                                                       /
************************************************************************

// Cwm_ChooseDressDlg diagnostics

#ifdef _DEBUG
void Cwm_ChooseDressDlg::AssertValid() const
{
    Cwm_ChooseDressBaseDlg::AssertValid();
}

void Cwm_ChooseDressDlg::Dump(CDumpContext& dc) const
{
    Cwm_ChooseDressBaseDlg::Dump(dc);
}

#endif //_DEBUG



/                                                                       /
************************************************************************

// Class definition for Class: Cwm_MDICHILDClientDlg
// Base Class                : Cwm_MDICHILDClientBaseDlg
// Derived from MFC Class    : CFormView
/                                                                       /
************************************************************************


class Cwm_MDICHILDClientDlg : public Cwm_MDICHILDClientBaseDlg

{
public:
    Cwm_MDICHILDClientDlg(LPCSTR lpszTemplateName);
    Cwm_MDICHILDClientDlg() : Cwm_MDICHILDClientBaseDlg() {};


#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
```

```
#endif

    virtual BOOL OnInitDialog();

    // YOU CAN ADD YOUR OWN CODE HERE


protected:
    //{{SIM_MSG_PROTO(Cwm_MDICHILDClientDlg)
    // DO NOT EDIT what you see in this block of generated
code.
    //}}SIM_MSG_PROTO(Cwm_MDICHILDClientDlg)

protected:
    virtual void DoDataExchange(CDataExchange* pDX);     //
DDX/DDV support

    DECLARE_DYNCREATE(Cwm_MDICHILDClientDlg)

    DECLARE_MESSAGE_MAP()

};


/                                                          /
************************************************************

//   C Startup function client area controls DRESSPICTURE
/                                                          /
************************************************************


CWnd* BLD_MDICHILDClientClFunc (CWnd* pWnd)
{
    Cwm_MDICHILDClientDlg*    pTheClientDlg;
    pTheClientDlg=new Cwm_MDICHILDClientDlg("DRESSPICTURE");

    if(!pTheClientDlg->Create(pWnd))
        {
         BLDDisplayMessage    (::GetActiveWindow    (),
BLD_CannotCreate,
            "DRESSPICTURE", MB_OK | MB_ICONHAND);
        return NULL;
        }
    return pTheClientDlg;
}
```

```
/                                                                    /
*********************************************************************

// Member Functions for Class: Cwm_MDICHILDClientDlg
// Base Class                : Cwm_MDICHILDClientBaseDlg
// Derived from MFC Class    : CFormView
/                                                                    /
*********************************************************************


Cwm_MDICHILDClientDlg::Cwm_MDICHILDClientDlg(LPCSTR
lpszTemplateName)
  : Cwm_MDICHILDClientBaseDlg(lpszTemplateName)
{
}



BOOL Cwm_MDICHILDClientDlg::OnInitDialog()
{
    return Cwm_MDICHILDClientBaseDlg::OnInitDialog();
}


IMPLEMENT_DYNCREATE(Cwm_MDICHILDClientDlg,
Cwm_MDICHILDClientBaseDlg)

BEGIN_MESSAGE_MAP(Cwm_MDICHILDClientDlg,
Cwm_MDICHILDClientBaseDlg)

    // YOU CAN ADD YOUR OWN CODE HERE

    //{{SIM_MSG_MAP(Cwm_MDICHILDClientDlg)
    // DO NOT EDIT what you see in this block of generated
code.
    //}}SIM_MSG_MAP(Cwm_MDICHILDClientDlg)

END_MESSAGE_MAP()


//{{SIM_MSG_BODY(Cwm_MDICHILDClientDlg)
// Message Handler Functions for you to edit will come here.



//}}SIM_MSG_BODY(Cwm_MDICHILDClientDlg)
```

```
void Cwm_MDICHILDClientDlg::DoDataExchange(CDataExchange* pDX)

{
    Cwm_MDICHILDClientBaseDlg::DoDataExchange(pDX);

    // YOU CAN ADD YOUR OWN CODE HERE


    //{{SIM_DATA_MAP(Cwm_MDICHILDClientDlg)
    // DO NOT EDIT what you see in this block of generated
code.

    //}}SIM_DATA_MAP
}



/                                                                    /
**********************************************************************

// Cwm_MDICHILDClientDlg diagnostics

#ifdef _DEBUG
void Cwm_MDICHILDClientDlg::AssertValid() const
{
    Cwm_MDICHILDClientBaseDlg::AssertValid();
}

void Cwm_MDICHILDClientDlg::Dump(CDumpContext& dc) const
{
    Cwm_MDICHILDClientBaseDlg::Dump(dc);
}

#endif //_DEBUG




/                                                                    /
**********************************************************************

// Class definition for Class: Cwm_PlateClientDlg
// Base Class                : Cwm_PlateClientBaseDlg
// Derived from MFC Class    : CFormView
/                                                                    /
**********************************************************************


class Cwm_PlateClientDlg : public Cwm_PlateClientBaseDlg
```

```
{
public:
    Cwm_PlateClientDlg(LPCSTR lpszTemplateName);
    Cwm_PlateClientDlg() : Cwm_PlateClientBaseDlg() {};

#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

    virtual BOOL OnInitDialog();

    // YOU CAN ADD YOUR OWN CODE HERE


protected:
    //{{SIM_MSG_PROTO(Cwm_PlateClientDlg)
    // DO NOT EDIT what you see in this block of generated
code.
    //}}SIM_MSG_PROTO(Cwm_PlateClientDlg)

protected:
    virtual void DoDataExchange(CDataExchange* pDX);    //
DDX/DDV support

    DECLARE_DYNCREATE(Cwm_PlateClientDlg)

    DECLARE_MESSAGE_MAP()

};


/                                                              /
**************************************************************

//   C Startup function client area controls PLATEPICTURE
/                                                              /
**************************************************************


CWnd* BLD_PlateClientClFunc (CWnd* pWnd)
{
    Cwm_PlateClientDlg*    pTheClientDlg;
    pTheClientDlg = new Cwm_PlateClientDlg("PLATEPICTURE");

    if(!pTheClientDlg->Create(pWnd))
        {
         BLDDisplayMessage    (::GetActiveWindow    (),
BLD_CannotCreate,
```

```
            "PLATEPICTURE", MB_OK | MB_ICONHAND);
        return NULL;
        }
    return pTheClientDlg;
}



/                                                                       /
************************************************************************

// Member Functions for Class: Cwm_PlateClientDlg
// Base Class                : Cwm_PlateClientBaseDlg
// Derived from MFC Class    : CFormView
/                                                                       /
************************************************************************


Cwm_PlateClientDlg::Cwm_PlateClientDlg(LPCSTR
lpszTemplateName)
  : Cwm_PlateClientBaseDlg(lpszTemplateName)
{
}



BOOL Cwm_PlateClientDlg::OnInitDialog()
{
    return Cwm_PlateClientBaseDlg::OnInitDialog();
}


IMPLEMENT_DYNCREATE(Cwm_PlateClientDlg,
Cwm_PlateClientBaseDlg)

BEGIN_MESSAGE_MAP(Cwm_PlateClientDlg,Cwm_PlateClientBaseDlg)


    // YOU CAN ADD YOUR OWN CODE HERE

    //{{SIM_MSG_MAP(Cwm_PlateClientDlg)
    // DO NOT EDIT what you see in this block of generated
code.
    //}}SIM_MSG_MAP(Cwm_PlateClientDlg)

END_MESSAGE_MAP()


//{{SIM_MSG_BODY(Cwm_PlateClientDlg)
```

```
// Message Handler Functions for you to edit will come here.


//}}SIM_MSG_BODY(Cwm_PlateClientDlg)


void  Cwm_PlateClientDlg::DoDataExchange(CDataExchange*  pDX)

{
    Cwm_PlateClientBaseDlg::DoDataExchange(pDX);

    // YOU CAN ADD YOUR OWN CODE HERE


    //{{SIM_DATA_MAP(Cwm_PlateClientDlg)
    // DO NOT EDIT what you see in this block of generated
code.

    //}}SIM_DATA_MAP
}



/                                                                    /
**********************************************************************

// Cwm_PlateClientDlg diagnostics

#ifdef _DEBUG
void Cwm_PlateClientDlg::AssertValid() const
{
    Cwm_PlateClientBaseDlg::AssertValid();
}

void Cwm_PlateClientDlg::Dump(CDumpContext& dc) const
{
    Cwm_PlateClientBaseDlg::Dump(dc);
}

#endif //_DEBUG



/                                                                    /
**********************************************************************
```

```
// Class definition for Class: Cwm_DesignClientDlg
// Base Class                : Cwm_DesignClientBaseDlg
// Derived from MFC Class    : CFormView
//                                                              /
*****************************************************************


class Cwm_DesignClientDlg : public Cwm_DesignClientBaseDlg
{
public:
    Cwm_DesignClientDlg(LPCSTR lpszTemplateName);
    Cwm_DesignClientDlg() : Cwm_DesignClientBaseDlg() {};

#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

    virtual BOOL OnInitDialog();

    // YOU CAN ADD YOUR OWN CODE HERE


protected:
    //{{SIM_MSG_PROTO(Cwm_DesignClientDlg)
    // DO NOT EDIT what you see in this block of generated
code.
    afx_msg void OnPaint();
    //}}SIM_MSG_PROTO(Cwm_DesignClientDlg)

protected:
    virtual  void  DoDataExchange(CDataExchange*  pDX);     //
DDX/DDV support

    DECLARE_DYNCREATE(Cwm_DesignClientDlg)

    DECLARE_MESSAGE_MAP()

};


/                                                              /
*****************************************************************

//   C Startup function client area controls DESIGNCLIENT
/                                                              /
*****************************************************************
```

```
CWnd* BLD_DesignClientClFunc (CWnd* pWnd)
{
    Cwm_DesignClientDlg*    pTheClientDlg;

     pTheClientDlg = new Cwm_DesignClientDlg("DESIGNCLIENT");

    if(!pTheClientDlg->Create(pWnd))
        {
         BLDDisplayMessage    (::GetActiveWindow    (),
BLD_CannotCreate,
            "DESIGNCLIENT", MB_OK | MB_ICONHAND);
        return NULL;
        }

           return pTheClientDlg;
}



/                                                                    /
**********************************************************************

// Member Functions for Class: Cwm_DesignClientDlg
// Base Class                : Cwm_DesignClientBaseDlg
// Derived from MFC Class    : CFormView
/                                                                    /
**********************************************************************


Cwm_DesignClientDlg::Cwm_DesignClientDlg(LPCSTR
lpszTemplateName)
  : Cwm_DesignClientBaseDlg(lpszTemplateName)
{
}



BOOL Cwm_DesignClientDlg::OnInitDialog()
{

    return Cwm_DesignClientBaseDlg::OnInitDialog();
}


IMPLEMENT_DYNCREATE(Cwm_DesignClientDlg,
Cwm_DesignClientBaseDlg)

BEGIN_MESSAGE_MAP(Cwm_DesignClientDlg,
Cwm_DesignClientBaseDlg)
```

```
    // YOU CAN ADD YOUR OWN CODE HERE

    //{{SIM_MSG_MAP(Cwm_DesignClientDlg)
    // DO NOT EDIT what you see in this block of generated
code.
    ON_WM_PAINT()
    //}}SIM_MSG_MAP(Cwm_DesignClientDlg)

END_MESSAGE_MAP()


//{{SIM_MSG_BODY(Cwm_DesignClientDlg)
// Message Handler Functions for you to edit will come here.









// MESSAGE HANDLER
void Cwm_DesignClientDlg::OnPaint()
{



     Cwm_DesignClientBaseDlg::OnPaint ();
}

//}}SIM_MSG_BODY(Cwm_DesignClientDlg)



void Cwm_DesignClientDlg::DoDataExchange(CDataExchange* pDX)

{
    Cwm_DesignClientBaseDlg::DoDataExchange(pDX);

    // YOU CAN ADD YOUR OWN CODE HERE


    //{{SIM_DATA_MAP(Cwm_DesignClientDlg)
    // DO NOT EDIT what you see in this block of generated
code.

    //}}SIM_DATA_MAP
```

```
}



/                                                                       /
**************************************************************

// Cwm_DesignClientDlg diagnostics

#ifdef _DEBUG
void Cwm_DesignClientDlg::AssertValid() const
{
    Cwm_DesignClientBaseDlg::AssertValid();
}

void Cwm_DesignClientDlg::Dump(CDumpContext& dc) const
{
    Cwm_DesignClientBaseDlg::Dump(dc);
}

#endif //_DEBUG



/                                                                       /
**************************************************************

// Class definition for Class: Cwm_ChoosePlateDlg
// Base Class                : Cwm_ChoosePlateBaseDlg
// Derived from MFC Class    : CDialog
/                                                                       /
**************************************************************


char PlateChoices [2][20] = {"apples", "ribbon"};

class Cwm_ChoosePlateDlg : public Cwm_ChoosePlateBaseDlg
{
public:
     Cwm_ChoosePlateDlg(LPCSTR   lpszTemplateName,CWnd
*pParentWnd);
    Cwm_ChoosePlateDlg() : Cwm_ChoosePlateBaseDlg() {};

#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
```

```
    virtual BOOL OnInitDialog();
    virtual void OnOK();
    virtual void OnCancel();

    // YOU CAN ADD YOUR OWN CODE HERE


protected:
    //{{SIM_MSG_PROTO(Cwm_ChoosePlateDlg)
    // DO NOT EDIT what you see in this block of generated
code.
    virtual BOOL ucPLATELISTInitdialog();
    void ucInitdialogPlate_Li(void);
    //}}SIM_MSG_PROTO(Cwm_ChoosePlateDlg)

protected:
    virtual void DoDataExchange(CDataExchange* pDX);     //
DDX/DDV support

    DECLARE_DYNCREATE(Cwm_ChoosePlateDlg)

    DECLARE_MESSAGE_MAP()

};


/                                                          /
************************************************************

//   C Startup function for modal dialogbox PLATELIST
/                                                          /
************************************************************


int BLD_ChoosePlateDlgFunc(CWnd *pWnd)
    {
    int iRet;
    Cwm_ChoosePlateDlg TheDlg ("PLATELIST",pWnd);

    iRet=TheDlg.DoModal();
    if( -1 == iRet )
        {
         BLDDisplayMessage    (::GetActiveWindow    (),
BLD_CannotCreate,
            "PLATELIST", MB_OK | MB_ICONHAND);
        }
    return iRet;
    }
```

```
/                                                                       /
************************************************************************

// Member Functions for Class: Cwm_ChoosePlateDlg
// Base Class                : Cwm_ChoosePlateBaseDlg
// Derived from MFC Class    : CDialog
/                                                                       /
************************************************************************


Cwm_ChoosePlateDlg::Cwm_ChoosePlateDlg(LPCSTR
lpszTemplateName,CWnd *pParentWnd)
  : Cwm_ChoosePlateBaseDlg(lpszTemplateName,pParentWnd)
{
}



BOOL Cwm_ChoosePlateDlg::OnInitDialog()
{
    return Cwm_ChoosePlateBaseDlg::OnInitDialog();
}



void Cwm_ChoosePlateDlg::OnOK()
{
    Cwm_ChoosePlateBaseDlg::OnOK();
}



void Cwm_ChoosePlateDlg::OnCancel()
{
    Cwm_ChoosePlateBaseDlg::OnCancel();
}


IMPLEMENT_DYNCREATE(Cwm_ChoosePlateDlg,
Cwm_ChoosePlateBaseDlg)

BEGIN_MESSAGE_MAP(Cwm_ChoosePlateDlg,Cwm_ChoosePlateBaseDlg)


    // YOU CAN ADD YOUR OWN CODE HERE
```

```
    //{{SIM_MSG_MAP(Cwm_ChoosePlateDlg)
    // DO NOT EDIT what you see in this block of generated
code.
    //}}SIM_MSG_MAP(Cwm_ChoosePlateDlg)

END_MESSAGE_MAP()


//{{SIM_MSG_BODY(Cwm_ChoosePlateDlg)
// Message Handler Functions for you to edit will come here.




// MESSAGE HANDLER
BOOL Cwm_ChoosePlateDlg::ucPLATELISTInitdialog()
{
     // Insert your code here
     return TRUE;
}


// CONTROL NOTIFICATION
void Cwm_ChoosePlateDlg::ucInitdialogPlate_Li( )
     {
     // Insert your code here

     int i;
       for(i=0;i<2;i++)
          SendDlgItemMessage(ID_Plate_List,LB_ADDSTRING,
                                   0,(LONG)PlateChoices[i]);

}

//}}SIM_MSG_BODY(Cwm_ChoosePlateDlg)



void  Cwm_ChoosePlateDlg::DoDataExchange(CDataExchange*  pDX)

{
    Cwm_ChoosePlateBaseDlg::DoDataExchange(pDX);

    // YOU CAN ADD YOUR OWN CODE HERE


    //{{SIM_DATA_MAP(Cwm_ChoosePlateDlg)
    // DO NOT EDIT what you see in this block of generated
```

```
code.

    //}}SIM_DATA_MAP
}



/                                                                       /
************************************************************************

// Cwm_ChoosePlateDlg diagnostics

#ifdef _DEBUG
void Cwm_ChoosePlateDlg::AssertValid() const
{
    Cwm_ChoosePlateBaseDlg::AssertValid();
}

void Cwm_ChoosePlateDlg::Dump(CDumpContext& dc) const
{
    Cwm_ChoosePlateBaseDlg::Dump(dc);
}

#endif //_DEBUG



// MENU AND CONTROL COMMAND
BOOL ucExit(CWnd *pWnd)
{
     // Insert your code here
     pWnd->SendMessage(WM_QUIT,0,0);
return FALSE;
}


// MENU AND CONTROL COMMAND
BOOL ucChangeColor(CWnd *pWnd)
{

     // Insert your code here


     static CHOOSECOLOR cc;
     static DWORD   dwCustColors[16];

     cc.lStructSize = sizeof(CHOOSECOLOR);
     cc.hwndOwner   = NULL;
     cc.hInstance   = NULL;
```

```
    cc.rgbResult    = RGB (0x80, 0x80, 0x80);
    cc.lpCustColors     = dwCustColors;
    cc.Flags        = CC_RGBINIT | CC_FULLOPEN;
    cc.lCustData    = 0L;
    cc.lpfnHook         = NULL;
    cc.lpTemplateName = NULL;

    return ChooseColor(&cc);
}


/                                                                       /
**************************************************************************

//   C Startup function for window merged
/                                                                       /
**************************************************************************


CWnd* BLD_mergedCreateWnd(CWnd *pAParent)
    {
    Cwm_mergedWnd *pmergedWnd;

    // Create the object
    pmergedWnd = new Cwm_mergedWnd();
    if (!pmergedWnd)
        return NULL;

    // Create the window
    pmergedWnd->SimInitWindow(pAParent);

    return pmergedWnd;
    }



/                                                                       /
**************************************************************************

// Member Functions for Window: Cwm_mergedWnd
// Base Class                 : Cwm_mergedBaseWnd
// Derived from MFC Class     : CWnd
/                                                                       /
**************************************************************************


Cwm_mergedWnd :: Cwm_mergedWnd()
    : Cwm_mergedBaseWnd()
{
```

```
}


Cwm_mergedWnd :: ~Cwm_mergedWnd()
{
}


IMPLEMENT_DYNCREATE(Cwm_mergedWnd,Cwm_mergedBaseWnd)

BEGIN_MESSAGE_MAP(Cwm_mergedWnd,Cwm_mergedBaseWnd)

    // YOU CAN ADD YOUR OWN CODE HERE

    //{{SIM_MSG_MAP(Cwm_mergedWnd)
    // DO NOT EDIT what you see in this block of generated
code.
    ON_WM_PAINT()
    //}}SIM_MSG_MAP(Cwm_mergedWnd)

END_MESSAGE_MAP()


//{{SIM_MSG_BODY(Cwm_mergedWnd)
// Message Handler Functions for you to edit will come here.




// MESSAGE HANDLER
void merger (CWnd*);
void Cwm_mergedWnd::OnPaint()
{
 merger(this);

     Cwm_mergedBaseWnd::OnPaint ();
}

//}}SIM_MSG_BODY(Cwm_mergedWnd)



/                                                                    /
**********************************************************************

// Cwm_mergedWnd diagnostics
```

```
#ifdef _DEBUG
void Cwm_mergedWnd::AssertValid() const
{
    Cwm_mergedBaseWnd::AssertValid();
}

void Cwm_mergedWnd::Dump(CDumpContext& dc) const
{
    Cwm_mergedBaseWnd::Dump(dc);
}

#endif //_DEBUG



/                                                                       /
************************************************************************

//   C Startup function for window merged
/                                                                       /
************************************************************************




/                                                                       /
************************************************************************

// Class definition for Class: Cwm_DESIGNERTBDlg
// Base Class                : Cwm_DESIGNERTBBaseDlg
// Derived from MFC Class    : CDialogBar
/                                                                       /
************************************************************************


class Cwm_DESIGNERTBDlg : public Cwm_DESIGNERTBBaseDlg
{
public:
    Cwm_DESIGNERTBDlg();
    virtual BOOL OnInitDialog();

#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif


    // YOU CAN ADD YOUR OWN CODE HERE
```

```
protected:
    //{{SIM_MSG_PROTO(Cwm_DESIGNERTBDlg)
    // DO NOT EDIT what you see in this block of generated
code.
    //}}SIM_MSG_PROTO(Cwm_DESIGNERTBDlg)

protected:

    virtual void DoDataExchange(CDataExchange* pDX);     //
DDX/DDV support

    DECLARE_DYNCREATE(Cwm_DESIGNERTBDlg)

    DECLARE_MESSAGE_MAP()

};


/                                                                    /
**********************************************************************

//   C Startup function toolbar TOPTOOLBAR
/                                                                    /
**********************************************************************


CWnd* BLD_DESIGNERTBTbFunc(CWnd *pWnd)
    {
    Cwm_DESIGNERTBDlg* pTheToolbar;

    pTheToolbar = new Cwm_DESIGNERTBDlg;


if(!pTheToolbar->Create(pWnd,"TOPTOOLBAR",BLDTOOLBARSTYLEDEF
AULT,1))
        {
         BLDDisplayMessage    (::GetActiveWindow    (),
BLD_CannotCreate,
            "TOPTOOLBAR", MB_OK | MB_ICONHAND);
        return NULL;
        }
    return (CWnd *)pTheToolbar;
    }



/                                                                    /
```

```
/////////////////////////////////////////////////////////////////////////

// Member Functions for Class: Cwm_DESIGNERTBDlg
// Base Class                : Cwm_DESIGNERTBBaseDlg
// Derived from MFC Class    : CDialogBar
//                                                                      /
/////////////////////////////////////////////////////////////////////////


Cwm_DESIGNERTBDlg::Cwm_DESIGNERTBDlg()
  : Cwm_DESIGNERTBBaseDlg()
{
}



BOOL Cwm_DESIGNERTBDlg::OnInitDialog()
{
    return Cwm_DESIGNERTBBaseDlg::OnInitDialog();
}


IMPLEMENT_DYNCREATE(Cwm_DESIGNERTBDlg,Cwm_DESIGNERTBBaseDlg)


BEGIN_MESSAGE_MAP(Cwm_DESIGNERTBDlg,  Cwm_DESIGNERTBBaseDlg)


    // YOU CAN ADD YOUR OWN CODE HERE

    //{{SIM_MSG_MAP(Cwm_DESIGNERTBDlg)
    // DO NOT EDIT what you see in this block of generated
code.
    //}}SIM_MSG_MAP(Cwm_DESIGNERTBDlg)

END_MESSAGE_MAP()


//{{SIM_MSG_BODY(Cwm_DESIGNERTBDlg)
// Message Handler Functions for you to edit will come here.



//}}SIM_MSG_BODY(Cwm_DESIGNERTBDlg)



void Cwm_DESIGNERTBDlg::DoDataExchange(CDataExchange* pDX)
{
```

```
    Cwm_DESIGNERTBBaseDlg::DoDataExchange(pDX);

    // YOU CAN ADD YOUR OWN CODE HERE


    //{{SIM_DATA_MAP(Cwm_DESIGNERTBDlg)
    // DO NOT EDIT what you see in this block of generated
code.

    //}}SIM_DATA_MAP
}


/                                                                    /
**********************************************************************

// Cwm_DESIGNERTBDlg diagnostics

#ifdef _DEBUG
void Cwm_DESIGNERTBDlg::AssertValid() const
{
    Cwm_DESIGNERTBBaseDlg::AssertValid();
}

void Cwm_DESIGNERTBDlg::Dump(CDumpContext& dc) const
{
    Cwm_DESIGNERTBBaseDlg::Dump(dc);
}

#endif //_DEBUG
```