Developing a Library of Human Computer Interface Patterns for Reuse

Loretta T. DiDonato, Ph. D. J. D., Villanova College of Engineering

ABSTRACT
When developing sailor systems requiring user interaction, it is efficient and currently desired to use, develop and integrate commonly used Human-Computer Interface (HCI) patterns to influence and enhance the tasks sailors perform on their systems and future complex systems being considered for NAVY legacy systems. In most development environments, designers developers, coders, testers refer to or integrate system controls based on usability guidelines. This paper elaborates on current research to provide re-useable human computer interfaces. For example, when developing new systems, a re-useable HCI pattern the user is already familiar with in determining display tasks would be developed in the context of a work domain. Such is considered in the work of Borchers called the Application Domain [1]. Command and Control is another domain being research by Stanard and Wampler. Osga suggests development of all domains to match using an Agile Process [3]. Van Welie research provides HCI patterns fit together interactively. This paper offers a short proposal for development of HCI patterns suitable to current development work using C# and the Microsoft Windows Presentation Format.

INTRODUCTION
Identifying the guidelines and applications of the work field and tasking which can be applied by automation aids requires a trained and experienced HCI designer in order to develop code in the user’s current environment, design mock up displays and provide both to the developer to create the GUI and add logic. HCI design patterns for re-usability drive the success of commonly used work patterns the user recognizes and has been trained to use at each workstation or console display. Task analysis of selected workstation production and controls to develop existing universal HCI design patterns into integrated commonality with existing systems and used for training on those systems and further development of automated training systems.

TEXT
The following code is sample Windows XML released in Windows Presentation Format C++. The pattern created follows which is a circle pattern and can be coded for further controls on an existing GUI with appropriate logic identified and implemented:

```cpp
#include <windows.h>
#include <d2d1.h>
#pragma comment(lib, "d2d1")
#include "basewin.h"

template <class T> void SafeRelease(T **ppT)
{
    if (*ppT)
    {
        (*ppT)->Release();
        *ppT = NULL;
    }
}
```
class MainWindow : public BaseWindow<MainWindow>
{
    ID2D1Factory *pFactory;
    ID2D1HwndRenderTarget *pRenderTarget;
    ID2D1SolidColorBrush *pBrush;
    D2D1_ELLIPSE ellipse;

    void CalculateLayout();
    HRESULT CreateGraphicsResources();
    void DiscardGraphicsResources();
    void OnPaint();
    void Resize();

public:

    MainWindow() : pFactory(NULL), pRenderTarget(NULL), pBrush(NULL)
    {
    }

    PCWSTR ClassName() const { return L"Circle Window Class"; }
    LRESULT HandleMessage(UINT uMsg, WPARAM wParam, LPARAM lParam);
};

// Recalculate drawing layout when the size of the window changes.

void MainWindow::CalculateLayout()
{
    if (pRenderTarget != NULL)
    {
        D2D1_SIZE_F size = pRenderTarget->GetSize();
        const float x = size.width / 2;
        const float y = size.height / 2;
        const float radius = min(x, y);
        ellipse = D2D1::Ellipse(D2D1::Point2F(x, y), radius, radius);
    }
}

HRESULT MainWindow::CreateGraphicsResources()
{
    HRESULT hr = S_OK;
    if (pRenderTarget == NULL)
    {
        RECT rc;
        GetClientRect(m_hwnd, &rc);
        D2D1_SIZE_U size = D2D1::SizeU(rc.right, rc.bottom);
        hr = pFactory->CreateHwndRenderTarget(
            D2D1::RenderTargetProperties(),
            D2D1::HwndRenderTargetProperties(m_hwnd, size),
            &pRenderTarget);
        if (SUCCEEDED(hr))
        {
            const D2D1_COLOR_F color = D2D1::ColorF(1.0f, 1.0f, 0);
            hr = pRenderTarget->CreateSolidColorBrush(color, &pBrush);
        }
    }
    return hr;
}

void MainWindow::DiscardGraphicsResources()
{
    // Code to release graphics resources
}
SafeRelease(&pRenderTarget);
SafeRelease(&pBrush);
}

void MainWindow::OnPaint()
{
    HRESULT hr =
    CreateGraphicsResources();
    if (SUCCEEDED(hr))
    {
        PAINTSTRUCT ps;
        BeginPaint(m_hwnd, &ps);

        pRenderTarget->BeginDraw();

        pRenderTarget->Clear(
            D2D1::ColorF(D2D1::ColorF::SkyBlue) );
        pRenderTarget->FillEllipse(ellipse,
            pBrush);

        hr = pRenderTarget->EndDraw();
        if (FAILED(hr) || hr ==
            D2DERR_RECREATE_TARGET)
        {
            DiscardGraphicsResources();
        }
        EndPaint(m_hwnd, &ps);
    }
}

void MainWindow::Resize()
{
    if (pRenderTarget != NULL)
    {
        RECT rc;
        GetClientRect(m_hwnd, &rc);

        D2D1_SIZE_U size =
        D2D1::SizeU(rc.right, rc.bottom);

        pRenderTarget->Resize(size);
        CalculateLayout();
        InvalidateRect(m_hwnd, NULL,
            FALSE);
    }
}

int WINAPI wWinMain(HINSTANCE
    hInstance, HINSTANCE, PWSTR, int
    nCmdShow)
{
    MainWindow win;

    if (!win.Create(L"Circle",
        WS_OVERLAPPEDWINDOW))
    {
        return 0;
    }

    ShowWindow(win.Window(),
        nCmdShow);
    // Run the message loop.

    MSG msg = { };
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return 0;
}

LRESULT
MainWindow::HandleMessage(UINT uMsg,
    WPARAM wParam, LPARAM lParam)
{
    switch (uMsg)
    {
        case WM_CREATE:
        {
            if (FAILED(D2D1CreateFactory(
                D2D1_FACTORY_TYPE_SINGLE_THREADED, &pFactory)))
            {
                return -1;  // Fail CreateWindowEx.
            }
            D2D1_FACTORY_TYPE_SINGLE_THREADED, &pFactory)))
            {
                return -1;  // Fail CreateWindowEx.
            }
            return 0;
        }
case WM_DESTROY:
    DiscardGraphicsResources();
    SafeRelease(&pFactory);
    PostQuitMessage(0);
    return 0;

case WM_PAINT:
    OnPaint();
    return 0;

case WM_SIZE:
    Resize();
    return 0;
}
return DefWindowProc(m_hwnd, uMsg, wParam, lParam);

Design guidelines would include decision analysis for spiral development (iterations of design and validation) or baseline configuration control board.

Console control/display configuration design concepts: such as, windows, menus, toolbars, text display, alarms, console features with common presentation need to be detailed.

A concept of operations as to how far and where the car will drive; such as, on what types of terrain or where the ship will sail with considerations for navigation depth and polar operations would drive design as well.

Console control/display evaluations would be decided by working together with customer liaison, preliminary task comparisons, performance factors and predicted performance results.

An individual console human performance evaluation derived from a simple design concept and initiated by a pilot study/user prototype would facilitate the design: common examples of these are:

- Human modeling studies
- Design metrics for human performance
- Human error avoidance
- Human machine interface design for usability study
- Human centered automation design

Hardware should then be selected using common enterprise display system and open architecture guidelines with a selected console ergonomics design.

**Figure 1: Developers create HCIs compatible with design**

**METHODOLOGY**

General design concepts provide the procedure design and simplification. A common control task, such as steering on a car or ship bridge, description would be a critical mission.
CONCLUSION

There are several design libraries of HCI patterns which are driven by standards such as MIL-STD-1472F, Department of Defense Design Criteria Standard for Human Engineering. Industry standards based on product development and system requirements should be developed to accommodate known and future use and modeled with training of system users.

REFERENCES


Stanard, Terry and Wampler, Jeffrey, Work-Centered HCI Design Patterns, (AFRL/HECS), Wright Patterson Air Force Base, Ohio.

Osga, Glenn, Matching HCI Design Patterns to Workflow within an Agile Process.


ACKNOWLEDGMENTS

The author acknowledges the professional courtesy and academic advisement of the Villanova College of Engineering and the support of Dr. Glen Osga and Dr. Terry Stanard who lead this research and implementation.

Dr. Loretta T. DiDonato, author, an employee of QED Systems Inc. holds a Ph.D. in Management Information Systems; she is currently a candidate for the MSCE at Villanova College of Engineering. Dr. DiDonato was the principal author of the ASNE, June 2003, Human Systems Integration Symposium technical paper, “Development and Implementation of a Sailor System Specification (S3) for DD(X)”. She was principal author and presenter of A Total Ship-Crew Model to Achieve Human Systems Integration at the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) December, 2004. Dr. DiDonato is working to develop a common HCI library pattern hierarchy development for users of systems and navigation control domain displays.